

Docket No.: 60188-721

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | | |
|---|---|------------------------|
| In re Application of | : | Customer Number: 20277 |
| | : | |
| Hidetoshi NARAHARA | : | Confirmation Number: |
| | : | |
| Serial No.: | : | Group Art Unit: |
| | : | |
| Filed: December 03, 2003 | : | Examiner: |
| | : | |
| For: SIMULATION METHOD AND EMULATION METHOD | : | |

**CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT**

Mail Stop CPD
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claim the priority of:

Japanese Patent Application No. JP2002-355819, filed on December 6, 2002.

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY


Michael E. Fogarty
Registration No. 36,139

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 MEF:gav
Facsimile: (202) 756-8087
Date: December 3, 2003

60188-1721
Hidetoshi NARAHARA
December 3, 2003

日 本 国 特 許 庁
JAPAN PATENT OFFICE

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: 2 0 0 2 年 1 2 月 6 日

出 願 番 号
Application Number: 特 願 2 0 0 2 - 3 5 5 8 1 9
[ST. 10/C]: [J P 2 0 0 2 - 3 5 5 8 1 9]

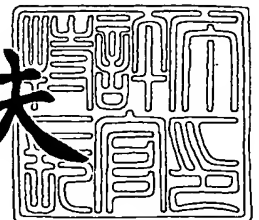
出 願 人
Applicant(s): 松下電器産業株式会社



2 0 0 3 年 1 0 月 2 8 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



出証番号 出証特 2 0 0 3 - 3 0 8 8 8 7 6



【書類名】 特許願

【整理番号】 5037540018

【提出日】 平成14年12月 6日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 17/50

【発明者】

 【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式
 会社内

 【氏名】 榎原 英俊

【特許出願人】

 【識別番号】 000005821

 【氏名又は名称】 松下電器産業株式会社

【代理人】

 【識別番号】 100077931

 【弁理士】

 【氏名又は名称】 前田 弘

【選任した代理人】

 【識別番号】 100094134

 【弁理士】

 【氏名又は名称】 小山 廣毅

【選任した代理人】

 【識別番号】 100110939

 【弁理士】

 【氏名又は名称】 竹内 宏

【選任した代理人】

 【識別番号】 100110940

 【弁理士】

 【氏名又は名称】 嶋田 高久

【選任した代理人】

【識別番号】 100113262

【弁理士】

【氏名又は名称】 竹内 祐二

【選任した代理人】

【識別番号】 100115059

【弁理士】

【氏名又は名称】 今江 克実

【選任した代理人】

【識別番号】 100115510

【弁理士】

【氏名又は名称】 手島 勝

【選任した代理人】

【識別番号】 100115691

【弁理士】

【氏名又は名称】 藤田 篤史

【手数料の表示】

【予納台帳番号】 014409

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0006010

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 シミュレーション方法およびエミュレーション方法

【特許請求の範囲】

【請求項1】 記述ブロック単位に実行履歴を記録する工程を有し、同一時刻に同じ記述ブロックが実行された場合、同一時刻の前回の実行履歴を消去する工程を有するシミュレーション方法。

【請求項2】 記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号のトレース結果から実行行を解析する工程を有するシミュレーション方法。

【請求項3】 記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号を単位時間単位にてトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号の単位時間単位のトレース結果から実行行を解析する工程を有するシミュレーション方法。

【請求項4】 記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をサイクル単位にてトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号のサイクル単位のトレース結果から実行行を解析する工程を有するシミュレーション方法。

【請求項5】 記述ブロック単位にその記述ブロックの入力に対応するハードウェアエミュレーション処理での信号を抽出する工程を有し、記述ブロックの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報を解析する工程を有し、エミュレーション実行処理工程にて、各記述ブロックの入力に対応するハードウェアエミュレーション処理での信号をトレースする工程を有し、前記記述ブロックの入力に対応するハードウェアエミュレ

ーション処理での信号の組み合わせと実行行の対応情報と前記記述ブロックの入力信号に対応するハードウェアエミュレーション処理での信号をトレースした結果から実行行を解析する工程を有するエミュレーション方法。

【請求項6】 ロジックコーン単位にそのロジックコーンの入力に対応するハードウェアエミュレーション処理での信号を抽出する工程を有し、ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報を解析する工程を有し、エミュレーション実行処理工程にて、各ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号をトレースする工程を有し、前記ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報と前記ロジックコーンの入力信号に対応するハードウェアエミュレーション処理での信号をトレースした結果から実行行を解析する工程を有するエミュレーション方法。

【請求項7】 記述ブロック単位にその記述ブロック内の各行が実行される入力条件を解析する工程を有し、前記入力条件と実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をトレースする工程を有し、前記入力条件と実行行の対応情報と前記記述ブロックの入力信号のトレース結果から、実行行を解析する工程を有するシミュレーション方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

この発明はシミュレーション方法およびエミュレーション方法に関し、さらに詳しくは、半導体素子設計時のハードウェア記述言語の検証方法に関する。

【0002】

【従来の技術】

近年、半導体素子の大規模化に伴い、半導体素子設計の効率化が重要視されている。このような背景の中、半導体素子の動作をプログラミング言語と同等に表現したハードウェア記述言語を用いて半導体素子を設計している。

【0003】

ハードウェア記述を用いた半導体素子の設計では、設計したハードウェア記述が半導体素子の仕様通りに動作するか検証しなければならない。この検証を行う従来のシミュレーション装置の概略構成を図32に示す。この装置は、シミュレーション実行処理部3と、実行行カウント処理部5とを備える。シミュレーション実行処理部3は、RTL記述1をテストベクタ2に基づいてシミュレーションする。RTL記述1は、半導体素子の機能をハードウェア記述言語で記述したものである。RTLとは、レジスタトランスファレベルの略で、記述の抽象度の一種である。テストベクタ2は、設計者が半導体素子の仕様に合わせて作成したものである。設計者は、実行処理部3によるシミュレーション結果4が仕様と一致しているか逐一確認しながら、RTL記述1が正しく設計されているか検証する。実行行カウント処理部5は、テストベクタ2の検証網羅性を測定するため、シミュレーション実行処理時のRTL記述1の実行行を逐一カウントし観測したカバレッジ結果6を出力する。設計者は、カバレッジ結果6からRTL記述1の未実行行を特定し、その未実行行が実行されるテストベクタ2を追加し再シミュレーションすることで検証漏れの無い品質の高い半導体素子を設計することができる。

【0004】

【特許文献1】

特許第2699377号公報

【0005】

【発明が解決しようとする課題】

図32に示したシミュレーション装置では、シミュレーション実行処理部3が逐次処理であるため、機能上実行されていない行が瞬間的に実行されてしまい、実行行カウント処理部5において誤カウントが生じ、不当に網羅性の高いカバレッジ結果6が出力されるという問題があった。その様子を図3(a)のRTL記述例と図3(b)のテストベクタ記述例を用いて説明する。

【0006】

図3(a)および(b)はハードウェア記述言語としてVerilog-HDLを用いた例であり、左端の番号は記述の行番号を意味する。図3(a)におい

て、100行および113行の `always` 文は、定義されている信号が変化する度に実行される構文であり、100行の場合、`a`、`b`、`c`のどれかが変化する度に101～110行が実行される。このような、ある信号変化に伴って一連の処理を行う記述のかたまりを記述ブロックと定義する。図3（a）の場合、100行から111行および113行から115行の2つの記述ブロックが存在する。

【0007】

時刻0において図3（b）の201行が実行され、`a`と`b`に0が代入される。次に`a`と`b`の変化を受けて図3（a）の100行と113行が評価されることになるのが逐次処理なので仮に100行から実行されたとする。この時、`a` = 0、`b` = 0であるが`c`の値が確定されていないため、`out`への代入は行われず、113行に処理が移り`c` = 1が代入される。次に`c`の変化を受けて再び100行が評価され、101行の`case`文によって $\{a, b, c\} = \{0, 0, 1\}$ の103行が実行され、`out`に0が代入される。次に時刻100において、図3（b）の202行が実行され、`a` = 1が代入される。次に`a`の変化を受け、図3（a）の100行が評価され、101行の`case`文によって $\{a, b, c\} = \{1, 0, 1\}$ の107行が実行され、`out`に1が代入される。次に113行に処理が移り`c` = 0が代入される。次にこの`c`の変化を受け、再び100行が評価され、101行の`case`文によって $\{a, b, c\} = \{1, 0, 0\}$ の106行が実行され、`out`に0が代入される。ここまでの処理におけるカバレッジ結果を図33に示す。図33は、図3（a）のRTL記述例の左端にカバレッジ結果として各行の実行回数を示したものである。上述のように逐次実行の度に実行行をカウントする場合には、機能的には実行されていない107行が実行されたことになり、不当に網羅性の高いカバレッジ結果になってしまう。

【0008】

この発明の目的は、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐシミュレーション方法を提供することである。

【0009】

【課題を解決するための手段】

この発明による第1のシミュレーション方法は、記述ブロック単位に実行行を記録し、同一時刻に同じ記述ブロックが実行された場合、同一時刻の前の実行履歴を消去する工程を有する。

【0010】

上記シミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐことができる。

【0011】

この発明による第2のシミュレーション方法は、記述ブロック単位にハードウェア記述を分解し、各記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号のトレース結果から実行行を解析する工程を有する。

【0012】

上記シミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定のオーバーヘッドを軽減することができる。

【0013】

この発明による第3のシミュレーション方法は、記述ブロック単位にハードウェア記述を分解し、各記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号を単位時間単位にてトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号の単位時間単位のトレース結果から実行行を解析する工程を有する。

【0014】

上記シミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐとともにオーバーヘッドを軽減することができる。

【0015】

この発明による第4のシミュレーション方法は、記述ブロック単位にハードウェア記述を分解し、各記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をサイクル単位にてトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号のサイクル単位のトレース結果から実行行を解析する工程を有する。

【0016】

上記シミュレーション方法によれば、設計する半導体素子の設計が完全同期設計である場合、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐシミュレーション方法とオーバーヘッドを軽減することができる。

【0017】

この発明の第1のエミュレーション方法は、記述ブロック単位にハードウェア記述を分解し、各記述ブロック単位にその記述ブロックの入力に対応するハードウェアエミュレーション処理での信号を抽出する工程を有し、記述ブロックの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報を解析する工程を有し、エミュレーション実行処理工程にて、各記述ブロックの入力に対応するハードウェアエミュレーション処理での信号をトレースする工程を有し、前記記述ブロックの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報と前記各記述ブロックの入力信号に対応するハードウェアエミュレーション処理での信号をトレースした結果から実行行を解析する工程を有する。

【0018】

上記エミュレーション方法によれば、ハードウェア記述言語のハードウェアエミュレーションにおけるカバレッジ測定を可能とすることができる。

【0019】

この発明による第2のエミュレーション方法は、ロジックコーン単位にハードウェア記述を分解し、各ロジックコーン単位にそのロジックコーンの入力に対応

するハードウェアエミュレーション処理での信号を抽出する工程を有し、ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報を解析する工程を有し、エミュレーション実行処理工程にて、各ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号をトレースする工程を有し、前記ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報と前記ロジックコーンの入力信号に対応するハードウェアエミュレーション処理での信号をトレースした結果から実行行を解析する工程を有する。

【0020】

上記エミュレーション方法によれば、ハードウェア記述言語のハードウェアエミュレーションにおいて、回路の最適化処理を実施した場合におけるカバレッジ測定を可能とする。

【0021】

この発明による第5のシミュレーション方法は、記述ブロック単位にハードウェア記述を分解し、各記述ブロック単位にその記述ブロック内の各行が実行される入力条件を解析する工程を有し、前記入力条件と実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をトレースする工程を有し、前記入力条件と実行行の対応情報と前記記述ブロックの入力信号のトレース結果から、実行行を解析する工程を有する。

【0022】

上記シミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおいて、入力信号の多い記述ブロックのカバレッジ測定が可能となる。

【0023】

【発明の実施の形態】

以下、この発明の実施の形態を図面を参照して詳しく説明する。なお、図中同一または相当部分には同一の符号を付しその説明は繰り返さない。

【0024】

(第1の実施形態)

第1の実施形態によるシミュレーション装置の構成を図1に示す。このシミュ

レーション装置は、シミュレーション実行処理部 3 と、実行行カウント処理部 10 とを備える。

【0025】

シミュレーション実行処理部 3 は、RTL 記述 1 をテストベクタ 2 に基づいてシミュレーションする。RTL 記述 1 は、半導体素子の機能をハードウェア記述言語で記述したものである。RTL とは、レジスタトランスファレベルの略で、記述の抽象度の一種である。テストベクタ 2 は、設計者が半導体素子の仕様に合わせて作成したものである。

【0026】

実行行カウント処理部 10 は、単位時間実行行トレース処理部 11 と、実行行集計処理部 12 とを含む。

【0027】

単位時間実行行トレース処理部 11 は、シミュレーション実行処理部 3 から実行行情報を受け取り、各記述ブロックについて単位時間ごとに実行行を抽出する。単位時間とは、設計者が指定できる最小時間を意味する。

【0028】

実行行集計処理部 12 は、単位時間実行行トレース処理部 11 により抽出された実行行情報に基づいて各行の実行回数をカウントし、シミュレーション実行処理終了後にカバレッジ結果 6 を出力する。

【0029】

次に、実行行カウント処理部 10 における処理について図 2 を参照しつつ説明する。

【0030】

シミュレーション実行処理部 3 によるシミュレーション実行処理の開始と同時に実行行カウント処理が開始される (ST101)。以後、シミュレーション実行処理が終了するまで実行行カウントループ ST102～ST111 が繰り返され、シミュレーション時刻が更新されるまで単位時間実行行トレースループ ST103～ST108 が繰り返される。

【0031】

ステップ S T 1 0 4 において、シミュレーション実行処理部 3 から現在の実行行情報を受け取り、シミュレータの実行行が変化したか調べる。変化していない場合はシミュレーション時刻が更新されるまで再びステップ S T 1 0 3 のループに戻る。ステップ S T 1 0 4 において実行行が変化していた場合、現在の実行行が記述ブロックの先頭行か調べる (S T 1 0 5)。記述ブロックの先頭行である場合、現在の実行行が含まれる記述ブロックのカウントテーブルをクリアする (S T 1 0 6)。カウントテーブルとは、各記述ブロック単位に単位時間ごとの実行履歴を記憶しておくものである。次に現在の実行行をカウントテーブルに登録する (S T 1 0 7)。ステップ S T 1 0 5 において記述ブロックの先頭行ではない場合は、カウントテーブルをクリアせずに、現在の実行行をカウントテーブルに登録する (S T 1 0 7)。次にシミュレーション時刻が更新されるまで再びステップ S T 1 0 3 のループに戻る。ここまでのステップ S T 1 0 3 ~ S T 1 0 8 における処理は単位時間実行行トレース処理部 1 1 において行われる。

【0032】

シミュレーション時刻が更新された場合、カウントテーブルに登録されている全ての実行行の実行行集計テーブルを +1 する (S T 1 0 9)。この処理は実行行集計処理 1 2 において行われる。実行行集計テーブルとは、各行についてその行の実行回数を記録するものである。次にステップ S T 1 1 0 においてカウントテーブルをクリアする。シミュレーション実行処理が終了するまでステップ S T 1 0 2 の実行行カウント処理ループに戻り、実行行カウント処理を繰り返す。

【0033】

図 3 (a) に示す R T L 記述 1 と図 3 (b) に示すテストベクタ 2 とに対して実行行カウント処理を行った場合のカバレッジ結果 6 を図 4 に示す。

【0034】

以上のように第 1 の実施形態では、単位時間実行行トレース処理部 1 1 により、単位時間単位に機能的に実行された行のみの実行行履歴を抽出するため、誤カウントの無い正確なカバレッジ結果 6 を得ることができる。

【0035】

(第 2 の実施形態)

図 3 2 に示した従来のシミュレーション装置ではシミュレーションの実行行をトレースする処理を実行行単位で行っているため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じ、シミュレーション速度が低下する。第 2 の実施形態では、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定のオーバーヘッドを軽減することを目的とする。

【 0 0 3 6 】

第 2 の実施形態によるシミュレーション装置の構成を図 5 に示す。このシミュレーション装置は、シミュレーション実行処理部 3 と、ブロック入力ー実行行対応表作成処理部 2 0 と、実行行カウント処理部 2 1 とを備える。

【 0 0 3 7 】

ブロック入力ー実行行対応表作成処理部 2 0 は、R T L 記述 1 を記述ブロック単位に分解し、各記述ブロックに対してブロックの入力信号の組み合わせと実行行の対応を解析し、そのブロックの入力信号の組み合わせと実行行の対応情報を抽出する。

【 0 0 3 8 】

実行行カウント処理部 2 1 は、ブロック入力トレース処理部 2 2 と、実行行解析処理部 2 3 と、実行行集計処理部 1 2 とを含む。

【 0 0 3 9 】

ブロック入力トレース処理部 2 2 は、ブロック入力ー実行行対応表作成処理部 2 0 により分解された記述ブロックの入力信号の変化毎に入力信号情報をシミュレーション実行処理部 3 から受け取る。

【 0 0 4 0 】

実行行解析処理部 2 3 は、ブロック入力トレース処理部 2 2 が受け取った入力信号情報と、ブロック入力ー実行行対応表作成処理部 2 0 が抽出したブロックの入力信号の組み合わせと実行行との対応情報とから実行行を抽出する。

【 0 0 4 1 】

実行行集計処理部 1 2 は、実行行解析処理部 2 3 により抽出された実行行情報を基に各行の実行回数をカウントし、シミュレーション実行処理終了後、カバレッジ結果 6 を出力する。

【0042】

次に、ブロック入力ー実行行対応表作成処理部20で行われる処理について図6を参照しつつ説明する。

【0043】

まず、RTL記述1を読み込む(ST201)。ここでは図3(a)に示したRTL記述が読み込まれるものとする。

【0044】

次に、読み込んだRTL記述1を記述ブロック単位に分割する(ST202)。この処理により、100行から110行までの記述ブロックと113行から115行までの記述ブロックとに分割される(図3(a)参照)。続くステップST203～ST209のブロック別対応表作成ループにおける処理は、ステップST202で分解された全ての記述ブロックに対して行われる。

【0045】

次に、処理対象の記述ブロックの入力信号を抽出する(ST204)。処理対象の記述ブロックが113～115行までの記述ブロックの場合、そのブロックの入力信号は、a, b, cになる(図3(a)参照)。次からのステップST205～ST208のブロック入力パターン生成ループにおける処理は、ステップST204で抽出されたブロックの入力信号の組み合わせ全てに対して行われる。113～115行までの記述ブロックの場合、 $\{a, b, c\} = \{0, 0, 0\} \sim \{1, 1, 1\}$ の8通りの組み合わせに対して処理されることになる。

【0046】

次に、対象組み合わせパターンでの対象記述ブロックの実行行を解析する(ST206)。対象パターンが $\{a, b, c\} = \{0, 0, 0\}$ の場合、実行行は100, 101, 102行となる(図3(a)参照)。

【0047】

次に、ステップST206で得られたブロック入力と実行行の関係をブロック入力ー実行行対応表に追加する(ST207)。

【0048】

次に、ブロック入力の組み合わせを変えて、ステップST205～ST208

のループに戻る。全ての組み合わせの処理が終了後、次の記述ブロックに処理を移し、ステップST203～ST209のループに戻る。

【0049】

以上の処理を図3（a）に示したRTL記述に対して行った場合のブロック入力ー実行行対応表を図7に示す。

【0050】

次に、実行行カウント処理部21における実行行カウント処理について図8を参照しつつ説明する。

【0051】

シミュレーション実行処理部3によるシミュレーション実行処理の開始と同時に実行行カウント処理を開始する（ST250）。

【0052】

次に、ブロック入力ー実行行対応表作成処理部20で得られた各ブロックの入力信号をシミュレーション実行処理におけるトレースポイントとして登録する（ST251）。

【0053】

次からのステップST252～ST256の実行行カウント処理ループはシミュレーション実行処理が終了するまで実行される。

【0054】

次に、シミュレーション実行処理においてトレースポイントが変化したか調べる（ST253）。ここまでの処理はブロック入力トレース処理部22において行われる。

【0055】

ステップST253においてトレースポイントが変化していない場合はそのままステップST252のループへ戻る。トレースポイントが変化していた場合は、そのトレースポイントの値とブロック入力ー実行行対応表作成処理部20から得られたブロック入力ー実行行対応表から実行行を抽出する（ST254）。この処理は実行行解析処理部23において行われる。

【0056】

次に、ステップ S T 2 5 4 で抽出された全ての実行行の実行行集計テーブルを +1 する (S T 2 5 5)。この処理は実行行集計処理部 1 2 において行われる。次にシミュレーション実行処理が終了するまで、ステップ S T 2 5 2 の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【0057】

図 3 (a) に示した R T L 記述 1 と図 3 (b) に示したテストベクタ 2 とに対して上述の実行行カウント処理を行った場合のカバレッジ結果 6 は図 3 3 に示した従来のカバレッジ結果と同等である。しかし、図 3 2 に示した従来のシミュレーション装置ではシミュレーションの実行行をトレースする処理を実行行単位で行っていたのに対し、第 2 の実施形態ではブロック入力トレース処理 2 2 により記述ブロックの入力信号のみをトレースするため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じシミュレーション速度が低下するという問題を解決できる。すなわち、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定のオーバーヘッドを軽減することができる。

【0058】

(第 3 の実施形態)

図 3 2 に示した従来のシミュレーション装置では、シミュレーション実行処理部 3 が逐次処理であるため、機能上実行されていない行が瞬間的に実行されてしまい、実行行カウント処理部 5 において誤カウントが生じ、不当に網羅性の高いカバレッジ結果 6 が出力されるおそれがある。また、シミュレーションの実行行をトレースする処理を実行行単位で行っているため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じ、シミュレーション速度が低下する。第 3 の実施形態では、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐとともにオーバーヘッドを軽減することを目的とする。

【0059】

第 3 の実施形態によるシミュレーション装置の構成を図 9 に示す。このシミュレーション装置は、シミュレーション実行処理部 3 と、ブロック入力-実行行対応表作成処理部 2 0 と、実行行カウント処理部 3 0 とを備える。

【0060】

実行行カウント処理部30は、単位時間ブロック入力トレース処理部31と、実行行解析処理部23と、実行行集計処理部12とを含む。

【0061】

単位時間ブロック入力トレース処理部31は、ブロック入力ー実行行対応表作成処理部20により分解された記述ブロックの入力信号の変化を単位時間毎に入力信号情報としてシミュレーション実行処理部3から受け取る。

【0062】

実行行解析処理部23は、単位時間ブロック入力トレース処理部31が受け取った入力信号情報とブロック入力ー実行行対応表作成処理部20が抽出したブロックの入力信号の組み合わせと実行行の対応情報から実行行を抽出する。

【0063】

次に、実行行カウント処理部30における処理について図10を参照しつつ説明する。

【0064】

シミュレーション実行処理部3によるシミュレーション実行処理の開始と同時に実行行カウント処理を開始する（ST300）。

【0065】

次に、ブロック入力ー実行行対応表作成処理部20で得られた各ブロックの入力信号をシミュレーション実行処理におけるトレースポイントとして登録する（ST301）。

【0066】

続くステップST302～ST310の実行行カウント処理ループはシミュレーション実行処理が終了するまで実行される。また、ステップST303～ST306の単位時間トレース処理ループはシミュレーション時刻が更新されるまで繰り返される。

【0067】

次に、シミュレーション実行処理においてトレースポイントが変化したか調べる（ST304）。変化していない場合はそのままステップST303のループ

に戻る。トレースポイントが変化した場合、変化したトレースポイントが含まれる記述ブロックの単位時間変化テーブルを更新し、ステップ S T 3 0 3 のループに戻る (S T 3 0 5)。単位時間変化テーブルとは、単位時間単位で各記述ブロックの入力信号の値を記憶するためのものである。

【0068】

シミュレーション時刻が更新された場合、各記述ブロックの単位時間変化テーブルが更新されたか調べる (S T 3 0 7)。ここまでの処理は単位時間ブロック入力トレース処理部 3 1 において行われる。

【0069】

ステップ S T 3 0 7 において単位時間変化テーブル更新されていない場合は、ステップ S T 3 0 2 のループに戻る。単位時間変化テーブルが更新された場合は、その更新された単位時間変化テーブルの値とブロック入力-実行行対応表作成処理部 2 0 から得られたブロック入力-実行行対応表から実行行を抽出する (S T 3 0 8)。この処理は実行行解析処理部 2 3 において行われる。

【0070】

次に、ステップ S T 3 0 8 で抽出された全ての実行行の実行行集計テーブルを +1 する (S T 3 0 9)。この処理は実行行集計処理部 1 2 において行われる。以後、シミュレーション実行処理が終了するまでステップ S T 3 0 2 の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【0071】

図 3 (a) に示した R T L 記述と図 3 (b) に示したテストベクタとに対して上述の実行行カウント処理を行った場合のカバレッジ結果 6 は図 4 に示したカバレッジ結果と同じである。

【0072】

以上のように第 3 の実施形態では、単位時間ブロック入力トレース処理部 3 1 により、単位時間単位に機能的に変化した記述ブロックの入力信号情報のみを抽出するため、誤カウントの無い正確なカバレッジ結果 6 を得ることができる。また、単位時間ブロック入力トレース処理部 3 1 により記述ブロックの入力信号のみをトレースするため、実行行が更新される度にシミュレーション処理に対して

オーバーヘッドが生じシミュレーション速度が低下するという問題を解決できる。すなわち、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定のオーバーヘッドを軽減することができる。

【0073】

(第4の実施形態)

図32に示した従来のシミュレーション装置では、シミュレーション実行処理部3が逐次処理であるため、機能上実行されていない行が瞬間的に実行されてしまい、実行行カウント処理部5において誤カウントが生じ、不当に網羅性の高いカバレッジ結果6が出力されるおそれがある。また、シミュレーションの実行行をトレースする処理を実行行単位で行っているため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じ、シミュレーション速度が低下する。第4の実施形態では、設計する半導体素子の設計が完全同期設計である場合に、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐとともにオーバーヘッドを軽減することを目的とする。

【0074】

第4の実施形態によるシミュレーション装置の構成を図11に示す。このシミュレーション装置は、シミュレーション実行処理部3と、ブロック入力ー実行行対応表作成処理部20と、実行行カウント処理部40とを備える。

【0075】

実行行カウント処理部40は、サイクル単位ブロック入力トレース処理部41と、実行行解析処理部23と、実行行集計処理部12とを含む。

【0076】

サイクル単位ブロック入力トレース処理部41は、ブロック入力ー実行行対応表作成処理部20により分解された記述ブロックの入力信号の変化をサイクル時間毎に入力信号情報としてシミュレーション実行処理部3から受け取る。

【0077】

実行行解析処理部23は、サイクル単位ブロック入力トレース処理部41が受け取った入力信号情報と、ブロック入力ー実行行対応表作成処理部20が抽出したブロックの入力信号の組み合わせと実行行の対応情報から実行行を抽出する。

【0078】

次に、実行行カウント処理部 4 0 における処理について図 1 2 を参照しつつ説明する。

【0079】

シミュレーション実行処理部 3 によるシミュレーション実行処理の開始と同時に実行行カウント処理を開始する (S T 4 0 0) 。

【0080】

次に、ブロック入力ー実行行対応表作成処理部 2 0 で得られた各ブロックの入力信号をシミュレーション実行処理におけるトレースポイントとして登録する (S T 4 0 1) 。

【0081】

次からのステップ S T 4 0 2 ～ S T 4 1 0 の実行行カウント処理ループはシミュレーション実行処理が終了するまで実行される。また、ステップ S T 4 0 3 ～ S T 4 0 6 の単位時間トレース処理ループはシミュレーション時刻が次のサイクルに更新されるまで繰り返される。

【0082】

次に、シミュレーション実行処理においてトレースポイントが変化したか調べる (S T 4 0 4) 。変化していない場合はそのままステップ S T 4 0 3 のループに戻る。トレースポイントが変化した場合は、変化したトレースポイントが含まれる記述ブロックのサイクル単位変化テーブルを更新し、ステップ S T 4 0 3 のループに戻る (S T 4 0 5) 。サイクル単位変化テーブルとは、サイクル単位で各記述ブロックの入力信号の値を記憶するためのものである。

【0083】

シミュレーション時刻が次のサイクルに更新された場合、各記述ブロックのサイクル単位変化テーブルが更新されたか調べる (S T 4 0 7) 。ここまでの処理はサイクル単位ブロック入力トレース処理部 4 1 において行われる。

【0084】

ステップ S T 4 0 7 においてサイクル単位変化テーブル更新されていない場合は、ステップ S T 4 0 2 のループに戻る。サイクル単位変化テーブルが更新された

場合は、その更新されたサイクル単位変化テーブルの値とブロック入力-実行行対応表作成処理部 20 から得られたブロック入力-実行行対応表から実行行を抽出する (ST408)。この処理は実行行解析処理部 23 において行われる。

【0085】

次に、ステップ ST408 で抽出された全ての実行行の実行行集計テーブルを +1 する (ST409)。この処理は実行行集計処理 12 において行われる。以後、シミュレーション実行処理が終了するまでステップ ST402 の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【0086】

図 3 (a) に示した RTL 記述と図 3 (b) に示したテストベクタとに対して上述の実行行カウント処理を行った場合のカバレッジ結果 6 は図 4 に示したカバレッジ結果と同じである。

【0087】

以上のように第 4 の実施形態では、サイクル単位ブロック入力トレース処理部 41 により、サイクル単位に機能的に変化した記述ブロックの入力信号情報のみを抽出するため、誤カウントの無い正確なカバレッジ結果 6 を得ることができる。また、サイクル単位ブロック入力トレース処理部 41 により記述ブロックの入力信号のみをトレースするため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じシミュレーション速度が低下するという問題を解決できる。また、実行行解析処理部 23 と実行行集計処理部 12 がサイクル単位でのみ実行となることから、第 3 の実施形態のシミュレーション方法と比べ、さらに処理速度を向上させることができる。

【0088】

(第 5 の実施形態)

ハードウェア記述を用いた半導体素子の設計では、設計したハードウェア記述が半導体素子の仕様通りに動作するか検証するため、ソフトウェアベースのシミュレータの代わりにハードウェアエミュレータを用いる場合がある。ハードウェアエミュレータはソフトウェアベースのシミュレータと比べ、1 万倍以上高速に検証することができる。次にハードウェアエミュレータの概要について説明する

。ハードウェアエミュレータでは、検証対象のRTL記述を論理構成処理によりゲートレベルのデータに変換し、それをFPGA等の実際のハードウェアに置き換えることで高速にRTL記述の動作をエミュレートすることができる。図3（a）に示したRTL記述の場合、図13に示すゲートレベルネットリストに置き換えられることになる。図13のゲートレベルネットリストにおいて、I1～I5はインスタンス名であり、個々の回路素子に付けられた固有の名称である。W1～W7は回路素子間を繋ぐノードに対して付けられた固有の名称である。しかしながら、エミュレーション処理では行実行という概念が全く無いことから、カバレッジを測定することができなかった。

【0089】

第5の実施形態では、ハードウェア記述言語のハードウェアエミュレーションにおけるカバレッジ測定を可能とすることを目的とする。

【0090】

第5の実施形態によるエミュレーション装置の構成を図14に示す。このエミュレーション装置は、論理合成部51と、エミュレーション実行処理部54と、ゲートレベルブロック入力ー実行行対応表作成処理部55と、実行行カウント処理部56とを備える。

【0091】

論理合成部51は、RTL記述1をハードウェアに置き換える為にゲートレベルネットリスト52に変換する。図3（a）に示したRTL記述の場合、図13のゲートレベルネットリストに変換される。また、ゲートレベルネットリスト変換時、RTL記述1とゲートレベルネットリスト52の対応を取るためのRTL-Gate対応情報53を出力する。図3（a）に示したRTL記述の場合のRTL-Gate対応情報を図15に示す。

【0092】

エミュレーション実行処理部54は、第1～第4の実施形態におけるシミュレーション実行処理部3に相当する機能を有する。

【0093】

ゲートレベルブロック入力ー実行行対応表作成処理部55は、RTL記述1を

記述ブロック単位に分解し、各記述ブロックに対してRTL-Gate対応情報53を用いて、エミュレーション実行処理部54でのブロックの入力信号の組み合わせと実行行の対応を解析し、そのエミュレーション実行処理部54でのブロックの入力信号の組み合わせと実行行の対応情報を抽出する。

【0094】

実行行カウント処理部56は、ゲートレベルブロック入力トレース処理部57と、実行行解析処理部23と、実行行集計処理部12とを含む。

【0095】

ゲートレベルブロック入力トレース処理部57は、ゲートレベルブロック入力-実行行対応表作成処理部55により分解された記述ブロックのエミュレーション実行処理部54での入力信号の変化情報をエミュレーション実行処理部54から受け取る。

【0096】

実行行解析処理部23は、ゲートレベルブロック入力トレース処理部57が受け取った入力信号情報と、ゲートレベルブロック入力-実行行対応表作成処理部55が抽出したエミュレーション実行処理部54でのブロックの入力信号の組み合わせと実行行との対応情報とから実行行を抽出する。

【0097】

実行行集計処理部12は、実行行解析処理部23より抽出した実行行情報を基に各行の実行回数をカウントし、エミュレーション実行処理終了後、カバレッジ結果6を出力する。

【0098】

次に、ゲートレベルブロック入力-実行行対応表作成処理部55における処理について図16を参照しつつ説明する。

【0099】

まずRTL記述1を読み込む(ST501)。ここでは図3(a)に示したRTL記述が読み込まれるものとする。

【0100】

次に、RTL-Gate対応情報53を読み込む(ST502)。

【0101】

次に、ステップST501で読み込んだRTL記述を記述ブロック単位に分割する(ST503)。この処理により、100行から110行までの記述ブロックと113行から115行までの記述ブロックとに分割される(図3(a)参照)。続くステップST504～ST511のブロック別対応表作成ループにおける処理は、ステップST503で分解された全ての記述ブロックに対して行われる。

【0102】

次に、処理対象の記述ブロックの入力信号を抽出する(ST505)。処理対象の記述ブロックが113行から115行までの記述ブロックの場合、そのブロックの入力信号は、a, b, cになる(図3(a)参照)。

【0103】

次に、ステップST505で抽出されたブロックの入力信号に対応するエミュレーション実行処理部54での信号を抽出する(ST506)。これ以降、ブロックの入力信号に対応するエミュレーション実行処理部54での信号のことを「ゲートレベルブロック入力」と呼ぶ。次からのステップST507～ST510のブロック入力パターン生成ループにおける処理は、ステップST505で抽出されたブロックの入力信号の組み合わせ全てに対して行われる。113行から115行までの記述ブロックの場合、 $\{a, b, c\} = \{W1, W2, W4\} = \{0, 0, 0\} \sim \{1, 1, 1\}$ の8通りの組み合わせに対して処理されることになる(図3(a)および図15参照)。

【0104】

次に、対象組み合わせパターンでの対象記述ブロックの実行行を解析する(ST508)。対象パターンが $\{a, b, c\} = \{W1, W2, W4\} = \{0, 0, 0\}$ の場合、実行行は100, 101, 102行となる。

【0105】

次に、ステップST508で得られたブロック入力と実行行の関係をゲートレベルブロック入力ー実行行対応表に追加する(ST509)。

【0106】

次に、ブロック入力の組み合わせを変えて、ステップ S T 5 0 7 ~ S T 5 1 0 のループに戻る。全ての組み合わせの処理が終了後、次の記述ブロックに処理を移し、ステップ S T 5 0 4 ~ S T 5 1 1 のループに戻る。

【 0 1 0 7 】

以上の処理を図 3 (a) に示した R T L 記述に対して行った場合のゲートレベルブロック入力-実行行対応表を図 1 7 に示す。

【 0 1 0 8 】

次に、実行行カウント処理部 5 6 における処理について図 1 8 を参照しつつ説明する。

【 0 1 0 9 】

エミュレーション実行処理部 5 4 によるエミュレーション実行処理の開始と同時に実行行カウント処理を開始する (S T 5 5 0) 。

【 0 1 1 0 】

次に、ゲートレベルブロック入力-実行行対応表作成処理部 5 5 で得られた各ブロックのゲートレベルブロック入力をエミュレーション実行処理におけるトレースポイントとして登録する (S T 5 5 1) 。次からのステップ S T 5 5 2 ~ S T 5 5 6 の実行行カウント処理ループはエミュレーション実行処理が終了するまで実行される。

【 0 1 1 1 】

次に、エミュレーション実行処理においてトレースポイントが変化したか調べる (S T 5 5 3) 。ここまでの処理はゲートレベルブロック入力トレース処理部 5 7 において行われる。

【 0 1 1 2 】

ステップ S T 5 5 3 においてトレースポイントが変化していない場合はそのままステップ S T 5 5 2 のループに戻る。変化した場合は、その変化したトレースポイントの値とゲートレベルブロック入力-実行行対応表作成処理部 5 5 から得られたゲートレベルブロック入力-実行行対応表から実行行を抽出する (S T 5 5 4) 。この処理は実行行解析処理部 2 3 において行われる。

【 0 1 1 3 】

次に、ステップ S T 5 5 4 で抽出された全ての実行行の実行行集計テーブルを + 1 する (S T 5 5 5) 。この処理は実行行集計処理 1 2 において行われる。

【 0 1 1 4 】

これ以降、エミュレーション実行処理が終了するまでステップ S T 5 5 2 の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【 0 1 1 5 】

図 3 (a) に示した R T L 記述と図 3 (b) に示したテストベクタに対して実行行カウント処理を行った場合のカバレッジ結果 6 は図 4 に示したカバレッジ結果と同じである。

【 0 1 1 6 】

以上のように第 5 の実施形態では、ゲートレベルブロック入力－実行行対応表作成処理部 5 5 から得られたゲートレベルブロック入力－実行行対応表とエミュレーション実行処理部 5 4 のブロック入力に対応する信号をトレースする処理により、R T L 記述 1 の実行行を推定することができ、従来のエミュレーション方法では測定できなかったカバレッジ観測を実現することができる。また、エミュレーション実行処理はソフトウェアシミュレーション処理と異なり逐次処理では無いため、誤カウントの無い正確なカバレッジ結果 6 を得ることができる。

【 0 1 1 7 】

(第 6 の実施形態)

図 1 4 に示したエミュレーション装置において、論理合成処理部 5 1 は R T L 記述 1 をゲートレベルネットリスト 5 2 に変換する。この際、ハードウェアに置き換える回路素子を少なくするため論理の最適化を行う場合がある。図 3 (a) に示した R T L 記述に対して最適化処理を行った場合のゲートレベルネットリストを図 1 9 に示す。この処理を行うことによりエミュレーション実行処理の対応回路規模を増加させることができ、エミュレーション処理速度も高速に行うことができる。しかしながら最適化処理を行うと、R T L 記述 1 の記述ブロックの入力に対応するゲートレベルブロック入力最適化処理により削除され、第 5 の実施形態で説明したエミュレーション方法を適用できない問題が生じる。第 6 の実施形態では、ハードウェア記述言語のハードウェアエミュレーションにおいて回

路の最適化処理を実施した場合におけるカバレッジ測定を可能とすることを目的とする。

【0118】

第6の実施形態によるエミュレーション装置の構成を図20に示す。このエミュレーション装置は、論理合成部51と、エミュレーション実行処理部54と、ロジックコーン入力-実行行対応表作成処理部61と、実行行カウント処理部62とを備える。

【0119】

論理合成処理部51は、RTL記述1をゲートレベルネットリスト52に変換する際、ゲートレベルネットリスト52の各ロジックコーンの入力に対応するRTL記述の信号情報（ロジックコーンRTL-Gate対応情報）60を出力する。ロジックコーンとは、論理回路をフリップフロップなどの記憶素子と組み合わせ回路に分割した際、各記憶素子の入力または回路の出力端子を起点として、その起点に影響を与える組み合わせ回路を意味する。図3（a）に示したRTL記述に対して論理合成処理を行った際のロジックコーンRTL-Gate対応情報60を図21に示す。

【0120】

ロジックコーン入力-実行行対応表作成処理部61は、ロジックコーンRTL-Gate対応情報60を用いて、RTL記述1をロジックコーン単位に分解し、各ロジックコーン単位にエミュレーション実行処理部54でのロジックコーン入力信号の組み合わせとRTL記述1の実行行の対応情報を抽出する。

【0121】

実行行カウント処理部62は、ロジックコーン入力トレース処理部63と、実行行解析処理部23と、実行行集計処理部12とを含む。

【0122】

ロジックコーン入力トレース処理部63は、エミュレーション実行処理部54でのロジックコーン入力の変化情報をエミュレーション実行処理部54から受け取る。

【0123】

実行行解析処理部 2 3 は、ロジックコーン入力トレース処理部 6 3 が受け取った入力信号情報と、ロジックコーン入力ー実行行対応表作成処理部 6 1 が抽出したエミュレーション実行処理部 5 4 でのロジックコーンの入力信号の組み合わせと実行行との対応情報とから実行行を抽出する。

【 0 1 2 4 】

実行行集計処理部 1 2 は、実行行解析処理部 2 3 より抽出した実行行情報を基に各行の実行回数をカウントし、エミュレーション実行処理終了後、カバレッジ結果 6 を出力する。

【 0 1 2 5 】

次に、ロジックコーン入力ー実行行対応表作成処理部 6 1 における処理について図 2 2 を参照しつつ説明する。

【 0 1 2 6 】

まず R T L 記述 1 を読み込む (S T 6 0 1) 。ここでは図 3 (a) に示した R T L 記述を読み込むものとする。

【 0 1 2 7 】

次に、ロジックコーン R T L - G a t e 対応情報 6 0 を読み込む (S T 6 0 2) 。

【 0 1 2 8 】

次に、ステップ S T 6 0 1 で読み込んだ R T L 記述 1 をロジックコーン単位に分割する (S T 6 0 3) 。図 3 (a) の R T L 記述 1 の場合、1 0 0 行から 1 1 5 行全てが 1 つのロジックコーンに相当する。

【 0 1 2 9 】

次からのステップ S T 6 0 4 ~ S T 6 1 1 のロジックコーン別対応表作成ループにおける処理は、ステップ S T 6 0 3 で分解された全てのロジックコーンに対して行われる。

【 0 1 3 0 】

次に、処理対象のロジックコーンの入力信号を抽出する (S T 6 0 5) 。図 3 (a) の R T L 記述 1 の場合、ロジックコーンの入力信号は、 a , b になる。

【 0 1 3 1 】

次に、ステップ S T 6 0 5 で抽出されたロジックコーンの入力信号に対応するエミュレーション実行処理部 5 4 での信号を抽出する (S T 6 0 6)。これ以降、ロジックコーン入力信号に対応するエミュレーション実行処理部 5 4 での信号のことを「ゲートレベルロジックコーン入力」と呼ぶ。

【0132】

次からのステップ S T 6 0 7 ~ S T 6 1 0 のロジックコーン入力パターン生成ループにおける処理は、ステップ S T 6 0 5 で抽出されたロジックコーンの入力信号の組み合わせ全てに対して行われる。図 3 (a) の R T L 記述 1 の場合、 $\{a, b\} = \{W1, W2\} = \{0, 0\} \sim \{1, 1\}$ の 4 通りの組み合わせに対して処理されることになる。

【0133】

次に、対象組み合わせパターンでの対象ロジックコーンの実行行を解析する (S T 6 0 8)。対象パターンが $\{a, b\} = \{W1, W2\} = \{0, 0\}$ の場合、実行行は 1 0 0, 1 0 1, 1 0 2, 1 1 3, 1 1 4 行となる。

【0134】

次に、ステップ S T 6 0 8 で得られたゲートレベルロジックコーン入力と実行行の関係をロジックコーン入力-実行行対応表に追加する (S T 6 0 9)。

【0135】

次に、ロジックコーンの入力の組み合わせを変えて、ステップ S T 6 0 7 ~ S T 6 1 0 のループに戻る。全ての組み合わせの処理が終了後、次のロジックコーンに処理を移し、ステップ S T 6 0 4 ~ S T 6 1 1 のループに戻る。以上の処理を図 3 (a) の R T L 記述 1 に対して行った場合のロジックコーン入力-実行行対応表を図 2 3 に示す。

【0136】

次に、実行行カウント処理部 6 2 における処理について図 2 4 を参照しつつ説明する。

【0137】

エミュレーション実行処理開始と同時に実行行カウント処理を開始する (S T 6 5 0)。

【0 1 3 8】

次に、ロジックコーン入力ー実行行対応表作成処理部 6 1 で得られた各ロジックコーンのゲートレベルロジックコーン入力をエミュレーション実行処理におけるトレースポイントとして登録する（S T 6 5 1）。

【0 1 3 9】

次からのステップ S T 6 5 2 ～ S T 6 5 6 の実行行カウント処理ループはエミュレーション実行処理が終了するまで実行される。

【0 1 4 0】

次に、エミュレーション実行処理においてトレースポイントが変化したか調べる（S T 6 5 3）。ここまでの処理はロジックコーン入力トレース処理 6 3 において行われる。

【0 1 4 1】

ステップ S T 6 5 3 においてトレースポイントが変化していない場合はそのままステップ S T 6 5 2 のループに戻る。変化した場合は、その変化したトレースポイントの値とロジックコーン入力ー実行行対応表作成処理部 6 1 から得られたロジックコーン入力ー実行行対応表から実行行を抽出する（S T 6 5 4）。この処理は実行行解析処理部 2 3 において行われる。

【0 1 4 2】

次に、ステップ S T 6 5 4 で抽出された全ての実行行の実行行集計テーブルを + 1 する（S T 6 5 5）。この処理は実行行集計処理部 1 2 において行われる。

【0 1 4 3】

これ以降、エミュレーション実行処理が終了するまでステップ S T 6 5 2 の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【0 1 4 4】

図 3（a）の R T L 記述 1 と図 3（b）のテストベクタとに対して実行行カウント処理を行った場合のカバレッジ結果 6 は図 4 に示したカバレッジ結果と同じである。

【0 1 4 5】

以上のように第 6 の実施形態では、ロジックコーンの入力、論理合成処理部

51による最適化処理を実行した場合でも必ずゲートレベルネットリスト52に存在するため、論理合成処理部51による最適化処理を実行した場合でもカバレッジを測定することができる。したがって、カバレッジを測定する場合でも最適化処理を実行でき、第5の実施形態のエミュレーション方法と比べ、大規模回路に対応することができる。また、エミュレーション処理速度も向上させることができる。

【0146】

(第7の実施形態)

第2～第4の実施形態によるシミュレーション装置では、ブロック入力ー実行行対応表作成処理部20は記述ブロック単位の各入力信号をbit単位で扱うため、記述ブロックの入力信号の数が多くなるとその入力信号全ての組み合わせが膨大な数になり処理できなくなるおそれがある。第7の実施形態では、ハードウェア記述言語のシミュレーションにおいて、入力信号の多い記述ブロックのカバレッジ測定を可能にすることを目的とする。

【0147】

第7の実施形態によるシミュレーション装置の構成を図25に示す。このシミュレーション装置は、シミュレーション実行処理部3と、条件別ブロック入力ー実行行対応表作成処理部70と、実行行カウント処理部71とを備える。

【0148】

条件別ブロック入力ー実行行対応表作成処理部70は、RTL記述1を記述ブロック単位に分解し、各記述ブロックに対してブロックの入力信号の条件と実行行の対応を解析し、そのブロックの入力信号の条件と実行行の対応情報を抽出する。第2～第4の実施形態におけるブロック入力ー実行行対応表作成処理部20ではそのブロックの入力をビット単位で扱っていたのに対し、条件別ブロック入力ー実行行対応表作成処理部70では各行の実行条件を解析し、ブロックの入力をその条件単位で扱う点が異なる。

【0149】

実行行カウント処理部71は、ブロック入力トレース処理部22と、条件別実行行解析処理部72と、実行行集計処理部12とを含む。

【0150】

ブロック入力トレース処理部22は、条件別ブロック入力ー実行行対応表作成処理部70により分解された記述ブロックの入力信号の変化毎に入力信号情報をシミュレーション実行処理部3から受け取る。

【0151】

条件別実行行解析処理部72は、ブロック入力トレース処理部22が受け取った入力信号情報と、条件別ブロック入力ー実行行対応表作成処理部70が抽出したブロックの入力条件と実行行の対応情報とから実行行を抽出する。

【0152】

実行行集計処理部12は、実行行解析処理部23により抽出された実行行情報を基に各行の実行回数をカウントし、シミュレーション実行処理終了後、カバレッジ結果6を出力する。

【0153】

次に、条件別ブロック入力ー実行行対応表作成処理部70における処理について図26を参照しつつ説明する。

【0154】

まずRTL記述1を読みこむ(ST701)。ここでは図27に示すRTL記述1を読み込むものとする。

【0155】

次に、読みこんだRTL記述1を記述ブロック単位に分割する(ST702)。この処理により、300行から304行までの記述ブロックと306行から308行までの記述ブロックとに分割される(図27参照)。

【0156】

次からのステップST703～ST710のブロック別対応表作成ループにおける処理は、ステップST702で分解された全ての記述ブロックに対して行われる。また、ステップST704～ST709の実行条件解析ループは、処理対象の記述ブロックの各行に対して解析が終了するまで繰り返される。

【0157】

次に、処理対象行の実行条件を解析する(ST705)。

【0 1 5 8】

次に、ステップ S T 7 0 5 で抽出した実行条件が、条件別ブロック入力ー実行行対応表に存在するか調べる (S T 7 0 6)。存在する場合は、条件別ブロック入力ー実行行対応表の同条件に現在の処理対象行を実行行として追加する (S T 7 0 7)。ステップ S T 7 0 6 において条件別ブロック入力ー実行行対応表に存在しない場合は、条件別ブロック入力ー実行行対応表に実行条件と実行行を追加する (S T 7 0 8)。

【0 1 5 9】

すべての記述ブロック内の行に対して処理を終了した後 (S T 7 0 9)、次の記述ブロックに対象ブロックを移し (S T 7 1 0)、ブロック別対応表作成ループ S T 7 0 3 に戻る。

【0 1 6 0】

以上の処理を図 2 7 の R T L 記述 1 に対して行った場合の条件別ブロック入力ー実行行対応表を図 2 8 に示す。

【0 1 6 1】

次に、実行行カウント処理部 7 1 における処理について図 2 9 を参照しつつ説明する。

【0 1 6 2】

シミュレーション実行処理開始と同時に実行行カウント処理を開始する (S T 7 5 0)。

【0 1 6 3】

次に、条件別ブロック入力ー実行行対応表作成処理部 7 0 で得られた各ブロックの入力信号をシミュレーション実行処理におけるトレースポイントとして登録する (S T 7 5 1)。

【0 1 6 4】

次からのステップ S T 7 5 2 ～ S T 7 5 6 の実行行カウント処理ループはシミュレーション実行処理が終了するまで実行される。

【0 1 6 5】

次に、シミュレーション実行処理においてトレースポイントが変化したか調べ

る (S T 7 5 3)。ここまでの処理はブロック入力トレース処理部 2 2 において行われる。

【0166】

ステップ S T 7 5 3 においてトレースポイントが変化していない場合はそのままステップ S T 7 5 2 のループへ戻る。トレースポイントが変化していた場合は、そのトレースポイントの値と条件別ブロック入力-実行行対応表作成処理部 7 0 から得られた条件別ブロック入力-実行行対応表から該当条件の実行行を抽出する (S T 7 5 4)。この処理は条件別実行行解析処理部 7 2 において行われる。

【0167】

次に、ステップ S T 7 5 4 で抽出された全ての実行行の実行行集計テーブルを + 1 する (S T 7 5 5)。この処理は実行行集計処理 1 2 において行われる。

【0168】

これ以降、シミュレーション実行処理が終了するまで、ステップ S T 7 5 2 の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【0169】

図 2 8 に示した R T L 記述 1 と図 3 0 に示すテストベクタ 2 とに対して実行行カウント処理を行った場合のカバレッジ結果 6 を図 3 1 に示す。

【0170】

第 7 の実施形態ではカバレッジの誤カウントを防止する処理を実施していない。したがってカバレッジ結果 (図 3 1) は誤カウントの含まれる結果となっている。しかしながら、第 3 ~ 第 4 の実施形態において説明したカバレッジの誤カウントを防止する処理を適用すれば誤カウントの無い正確なカバレッジ結果を得ることができる。

【0171】

第 2 ~ 第 4 の実施形態では記述ブロック入力と実行行の対応を b i t 単位で扱っているのに対し、第 7 の実施形態では条件別ブロック入力実行行対応表作成処理部 7 0 が記述ブロックの各行の実行条件単位で入力条件と実行行の対応をとるため、入力信号の多い記述ブロックに対してカバレッジ測定が可能となる。

【 0 1 7 2 】

【発明の効果】

この発明によるシミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐことができる。

【図面の簡単な説明】

【図 1】 第 1 の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図 2】 図 1 に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図 3】 (a) は R T L 記述の一例を示す図であり、(b) はテストベクタの一例を示す図である。

【図 4】 カバレッジ結果を示す図である。

【図 5】 第 2 の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図 6】 図 5 に示したブロック入力ー実行行対応表作成処理部における処理の流れを示すフローチャートである。

【図 7】 ブロック入力ー実行行対応表の一例を示す図である。

【図 8】 図 5 に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図 9】 第 3 の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図 1 0】 図 9 に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図 1 1】 第 4 の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図 1 2】 図 1 1 に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図 1 3】 ゲートレベルネットリストの一例を示す図である。

【図 1 4】 第 5 の実施形態によるエミュレーション装置の構成を示すプロ

ック図である。

【図 1 5】 R T L - G a t e 対応情報の一例を示す図である。

【図 1 6】 図 1 4 に示したゲートレベルブロック入力ー実行行対応表作成処理部における処理の流れを示すフローチャートである。

【図 1 7】 ゲートレベルブロック入力ー実行行対応表の一例を示す図である。

【図 1 8】 図 1 4 に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図 1 9】 ゲートレベルネットリストの一例を示す図である。

【図 2 0】 第 6 の実施形態によるエミュレーション装置の構成を示すブロック図である。

【図 2 1】 ロジックコーン R T L - G a t e 対応情報の一例を示す図である。

【図 2 2】 図 2 0 に示したロジックコーン入力ー実行行対応表作成処理部における処理の流れを示すフローチャートである。

【図 2 3】 ロジックコーン入力ー実行行対応表の一例を示す図である。

【図 2 4】 図 2 0 に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図 2 5】 第 7 の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図 2 6】 図 2 5 に示した条件別ブロック入力ー実行行対応表作成処理部における処理の流れを示すフローチャートである。

【図 2 7】 R T L 記述の一例を示す図である。

【図 2 8】 条件別ブロック入力ー実行行対応表の一例を示す図である。

【図 2 9】 図 2 5 に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図 3 0】 テストベクタの一例を示す図である。

【図 3 1】 カバレッジ結果を示す図である。

【図 3 2】 従来のシミュレーション装置の構成を示すブロック図である。

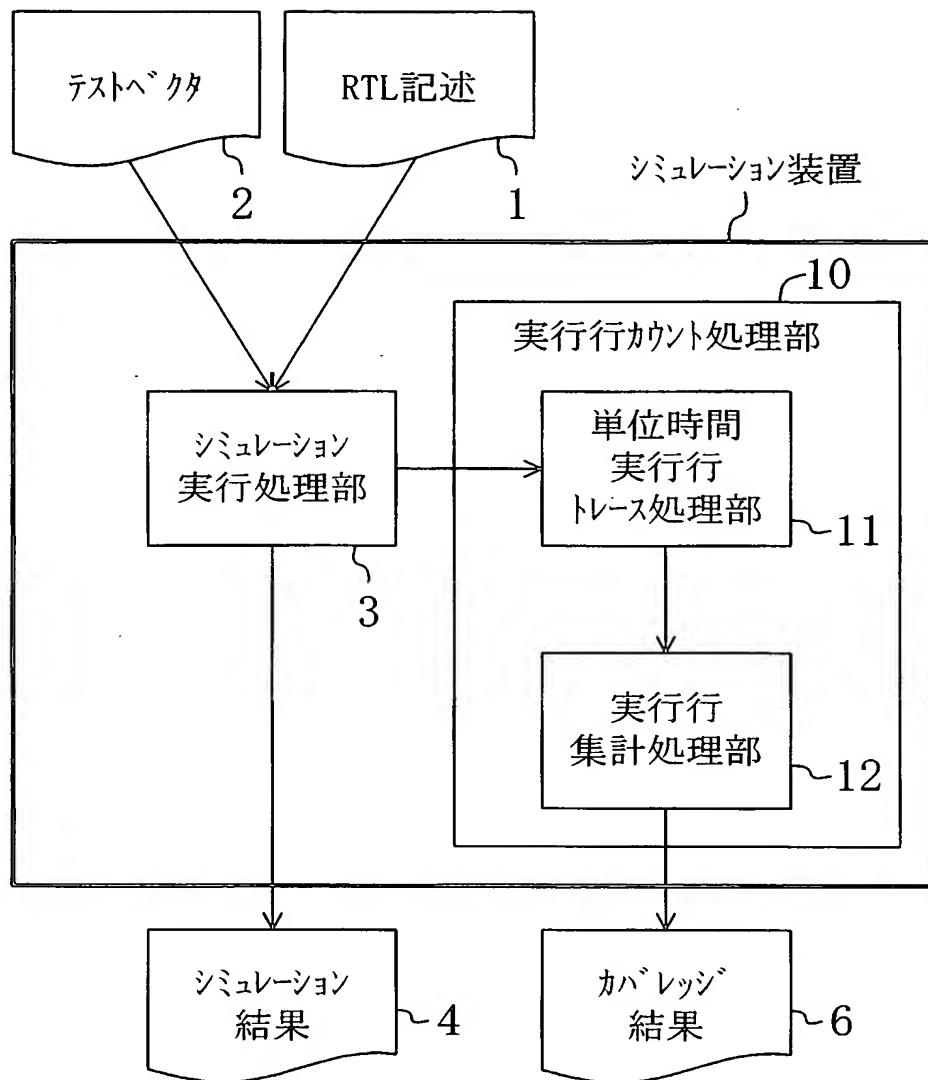
【図 3 3】 カバレッジ結果を示す図である。

【符号の説明】

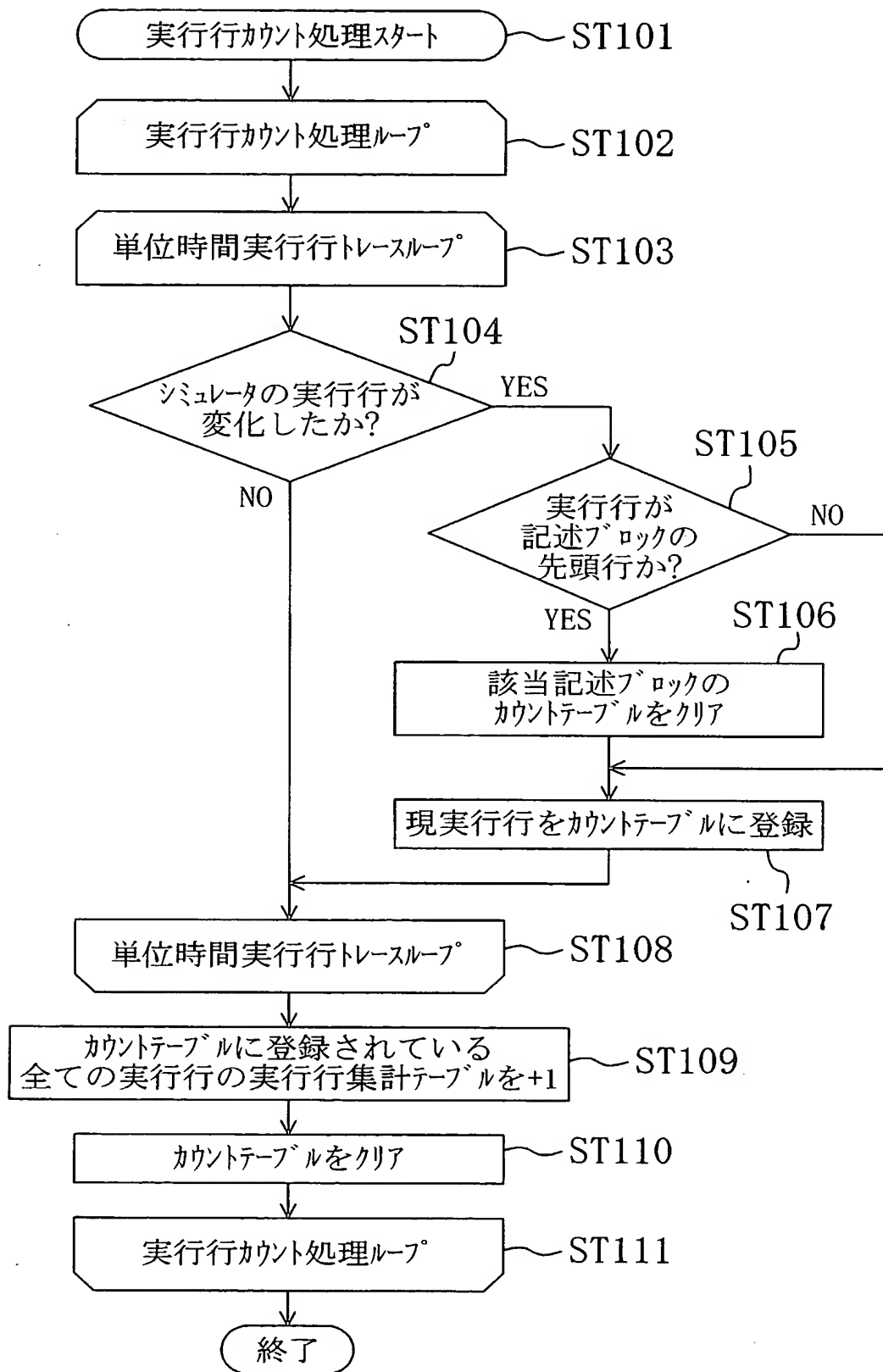
1 R T L 記述、2 テストベクタ、3 シミュレーション実行処理部、4 シミュレーション結果、5, 1 0, 2 1, 3 0, 4 0, 5 6, 6 2, 7 1 実行行カウント処理部、6 カバレッジ結果、1 1 単位時間実行行トレース処理部、1 2 実行行集計処理部、2 0 ブロック入力ー実行行対応表作成処理部、2 2 ブロック入力トレース処理部、2 3 実行行解析処理部、3 1 単位時間ブロック入力トレース処理部、4 1 サイクル単位ブロック入力トレース処理部、5 1 論理合成処理部、5 2 ゲートレベルネットリスト、5 3 R T L - G a t e 対応情報、5 4 エミュレーション実行処理部、5 5 ゲートレベルブロック入力ー実行行対応表作成処理部、5 7 ゲートレベルブロック入力トレース処理部、6 0 ロジックコーン R T L - G a t e 対応情報、6 1 ロジックコーン入力ー実行行対応表作成処理部、6 3 ロジックコーン入力トレース処理部、7 0 条件別ブロック入力ー実行行対応表作成処理部、7 2 条件別実行行解析処理部。

【書類名】 図面

【図 1】



【図 2】



【図 3】

(a)

1
}

```
100:    always @ (a or b or c) begin
101:        case({a, b, c})
102:            3' b000:out=0;
103:            3' b001:out=0;
104:            3' b010:out=0;
105:            3' b011:out=0;
106:            3' b100:out=0;
107:            3' b101:out=1;
108:            3' b110:out=1;
109:            3' b111:out=1;
110:        endcase
111:    end
112:
113:    always @ (a) begin
114:        c=~a;
115:    end
```

(b)

2
}

```
200:    initial begin
201:        #0      a=0; b=0;
202:        #100    a=1;
203:    end
```

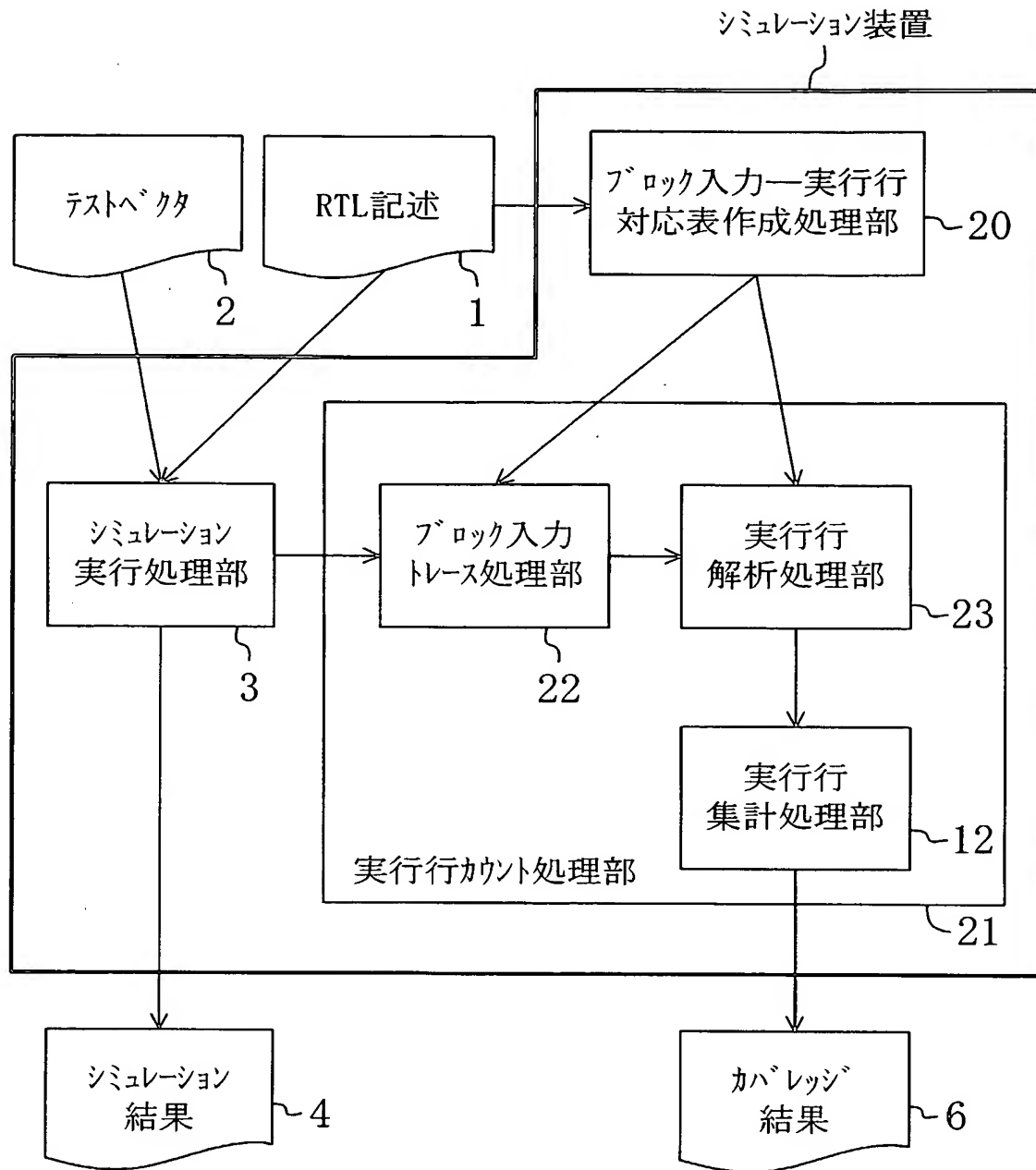
【図 4】

6
}

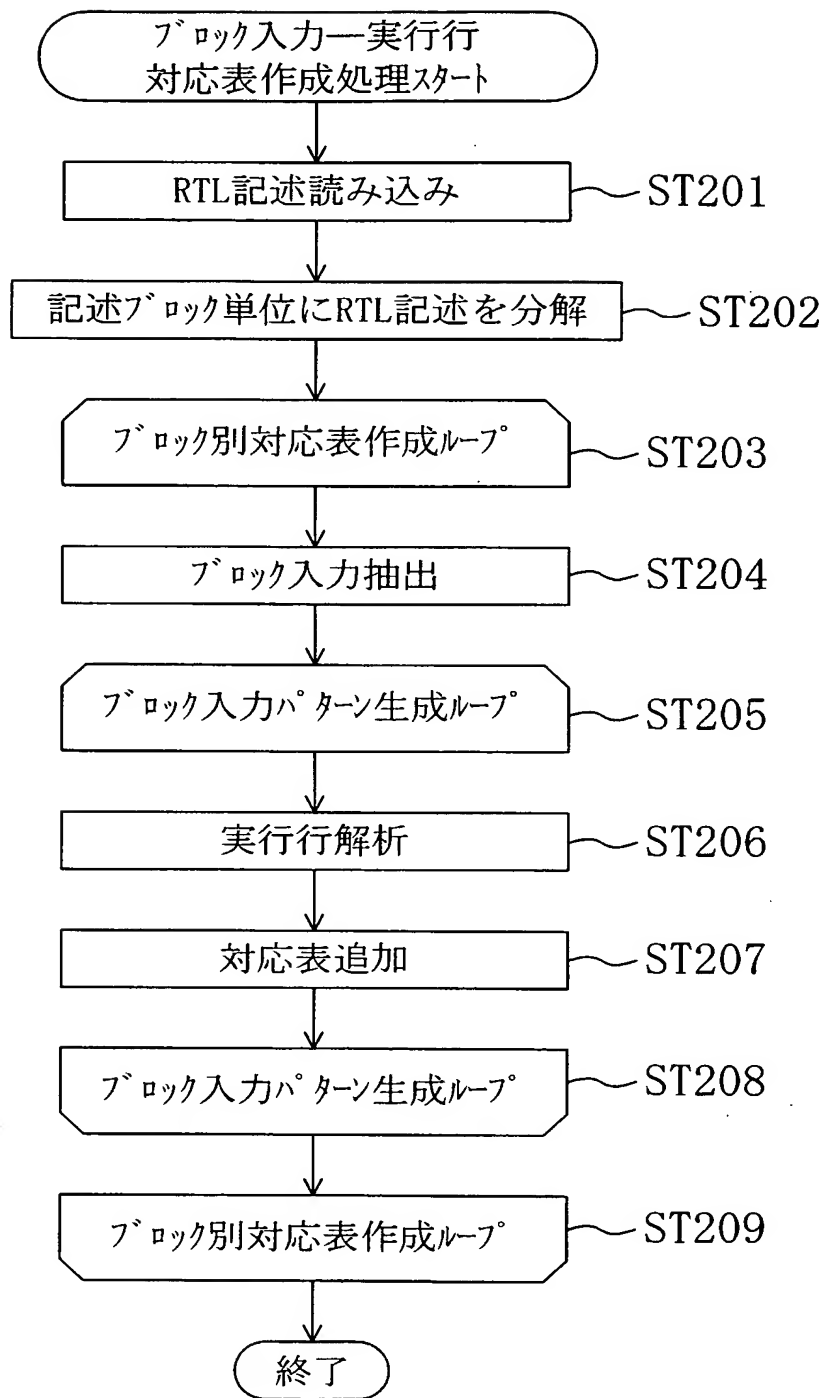
実行回数

```
2    100:    always @ (a or b or c) begin
2    101:                case({a,b,c})
        102:                    3' b000:out=0;
1    103:                    3' b001:out=0;
        104:                    3' b010:out=0;
        105:                    3' b011:out=0;
1    106:                    3' b100:out=0;
        107:                    3' b101:out=1;
        108:                    3' b110:out=1;
        109:                    3' b111:out=1;
        110:                endcase
        111:    end
        112:
2    113:    always @ (a) begin
2    114:                c=~a;
        115:    end
```

【図 5】



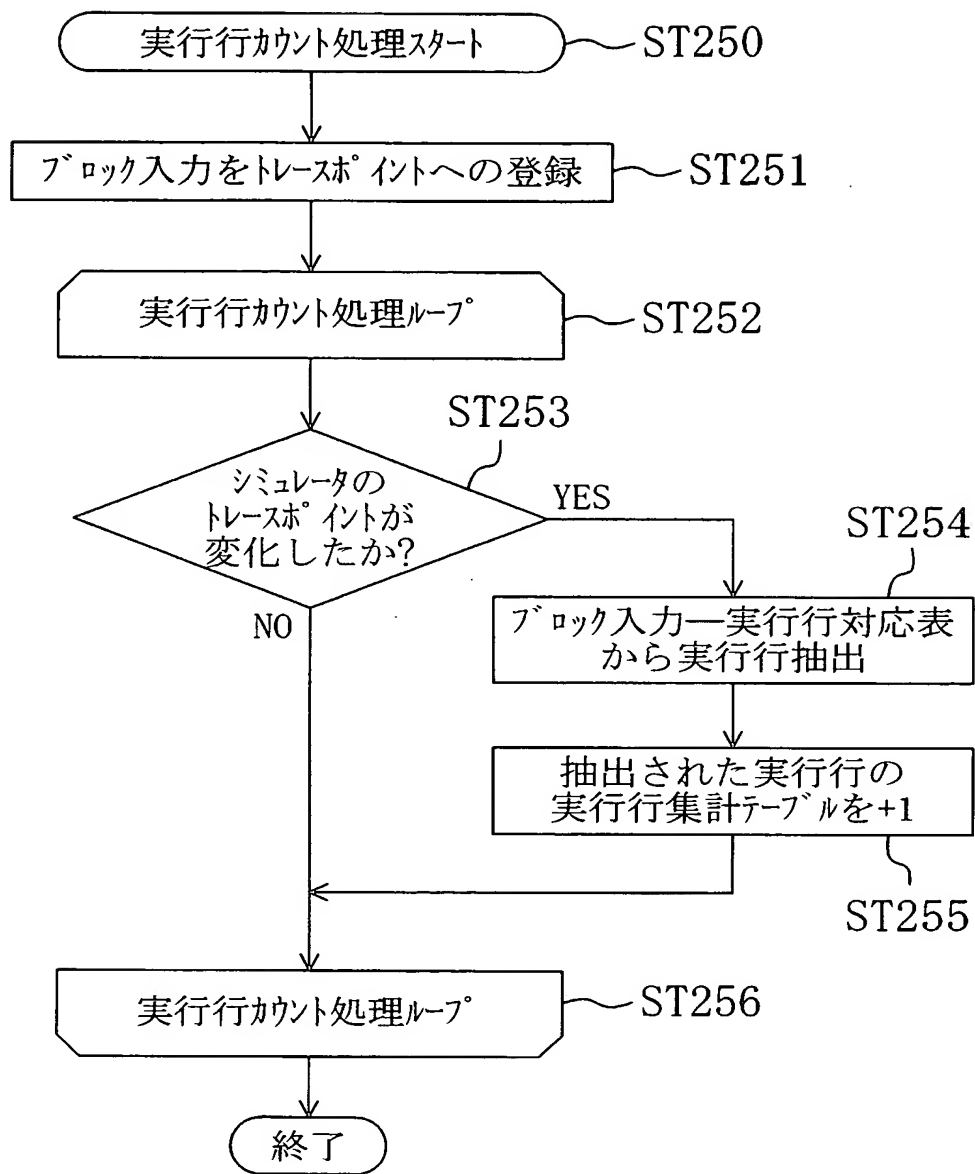
【図 6】



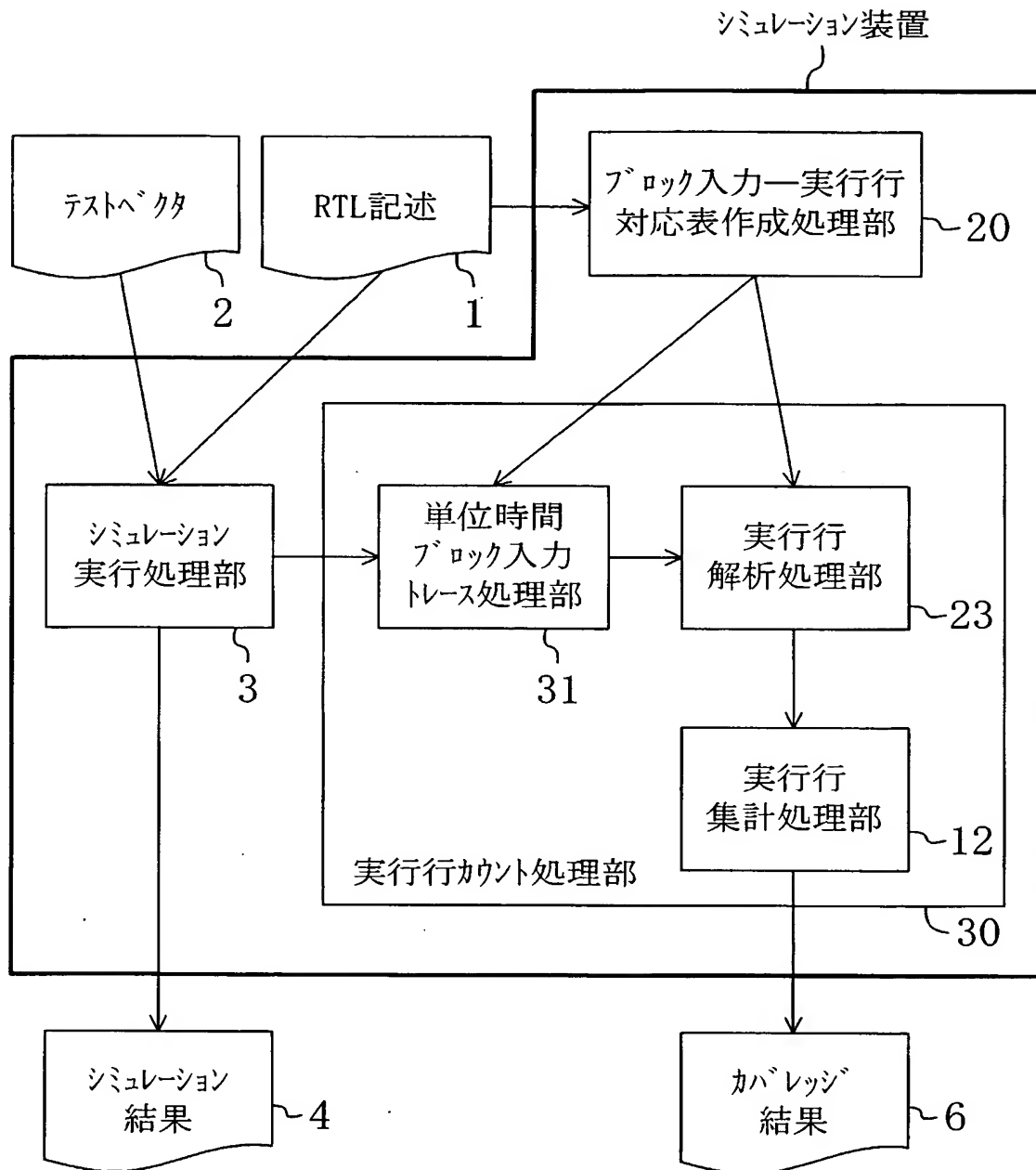
【図 7】

| 100-110行の記述ブロック | |
|-----------------|---------------|
| abc | 実行行 |
| 000 | 100, 101, 102 |
| 001 | 100, 101, 103 |
| 010 | 100, 101, 104 |
| 011 | 100, 101, 105 |
| 100 | 100, 101, 106 |
| 101 | 100, 101, 107 |
| 110 | 100, 101, 108 |
| 111 | 100, 101, 109 |
| 113-115行の記述ブロック | |
| a | 実行行 |
| 0 | 113, 114 |
| 1 | 113, 114 |

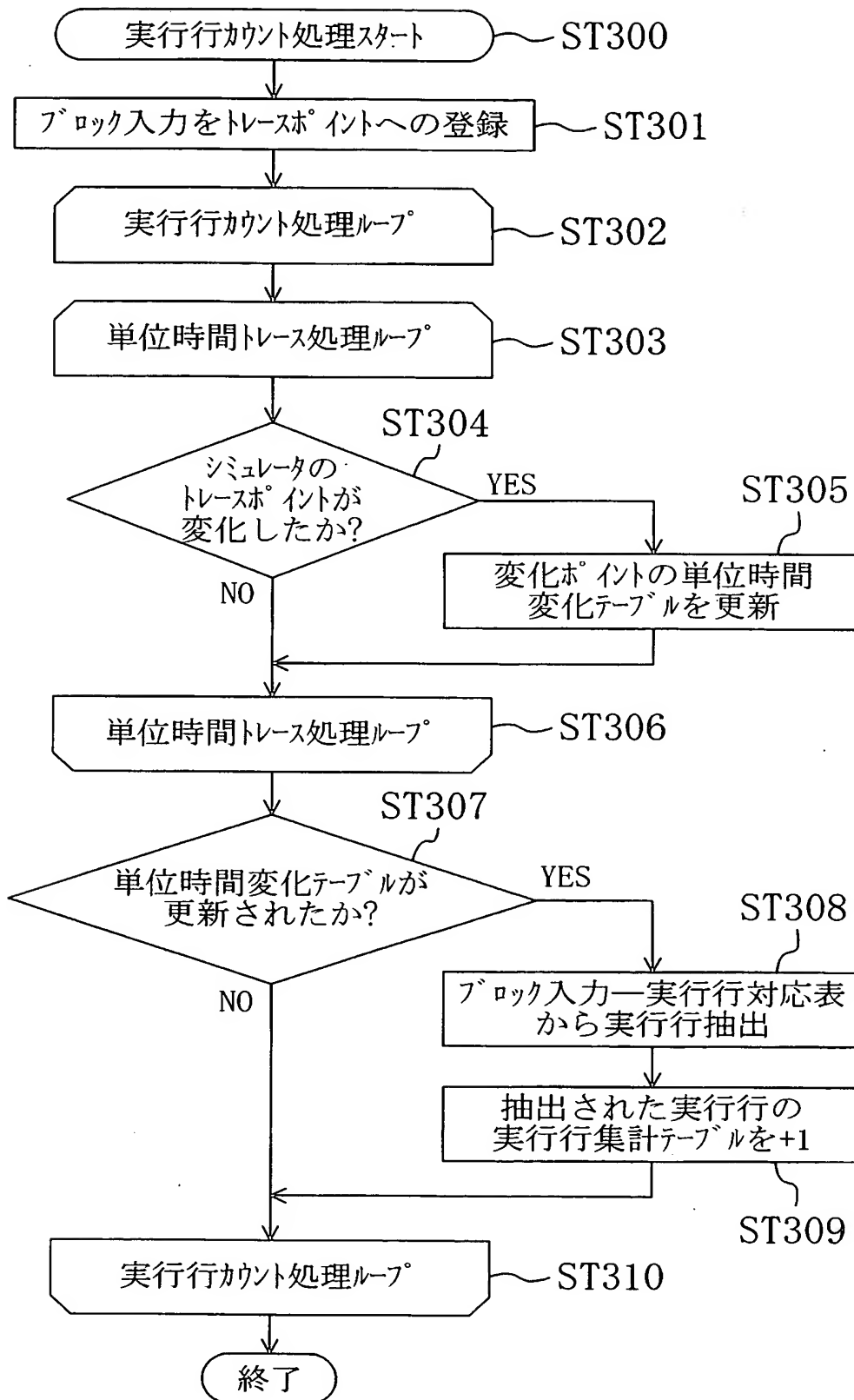
【図 8】



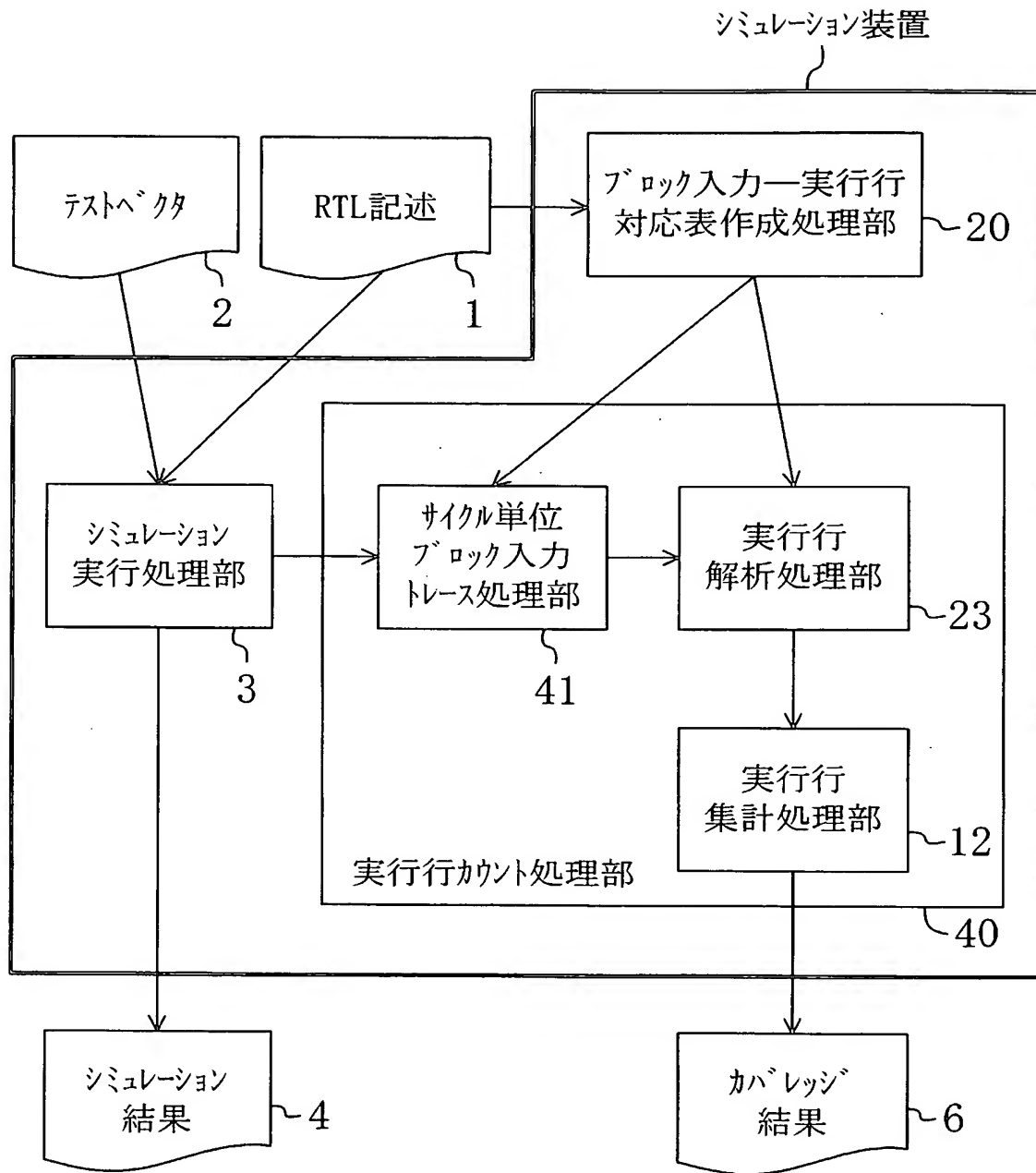
【図 9】



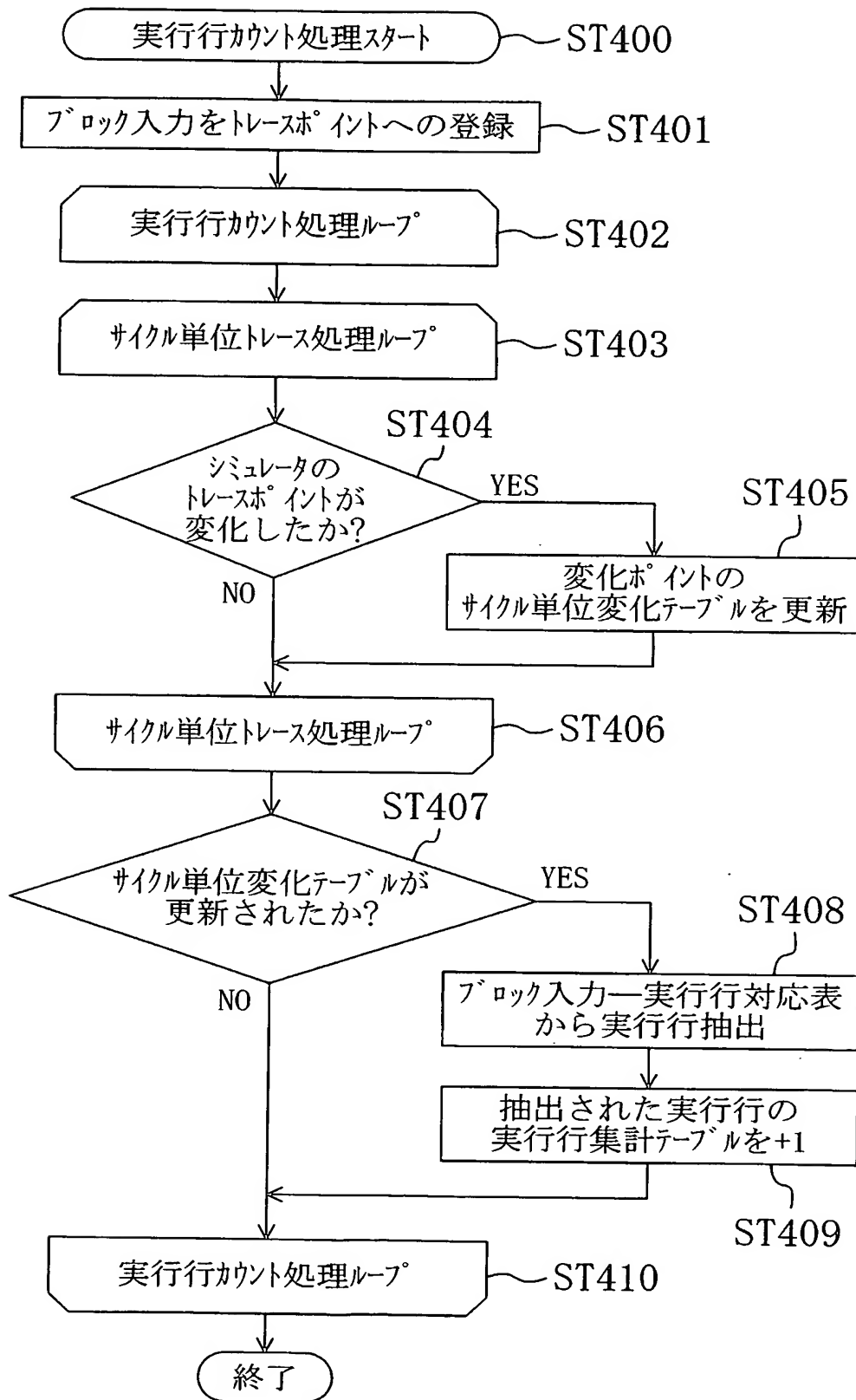
【図10】



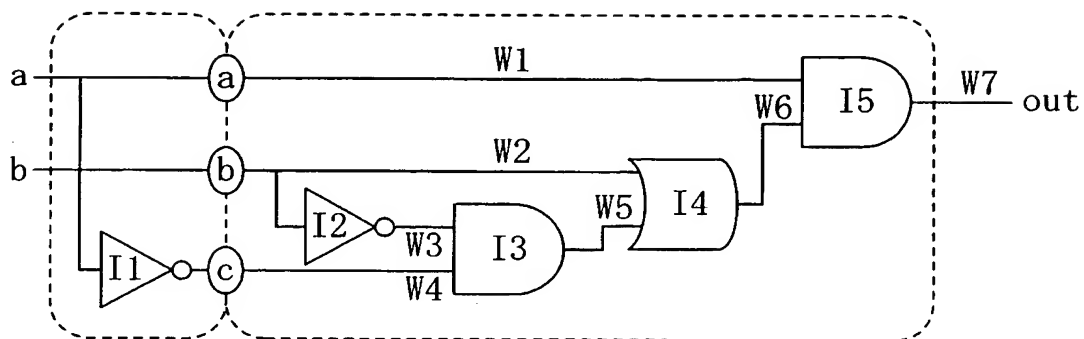
【図 11】



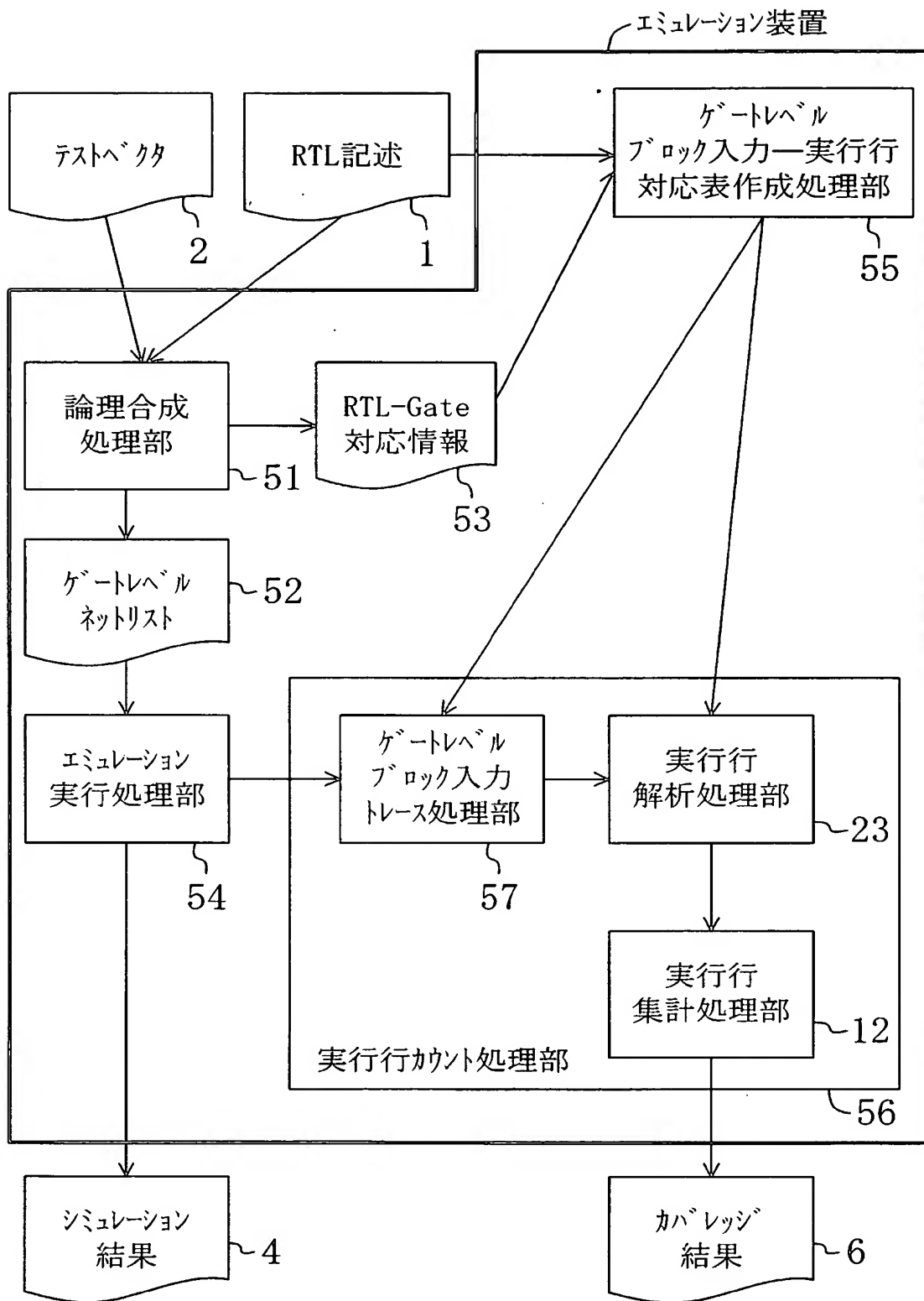
【図 12】



【図 13】



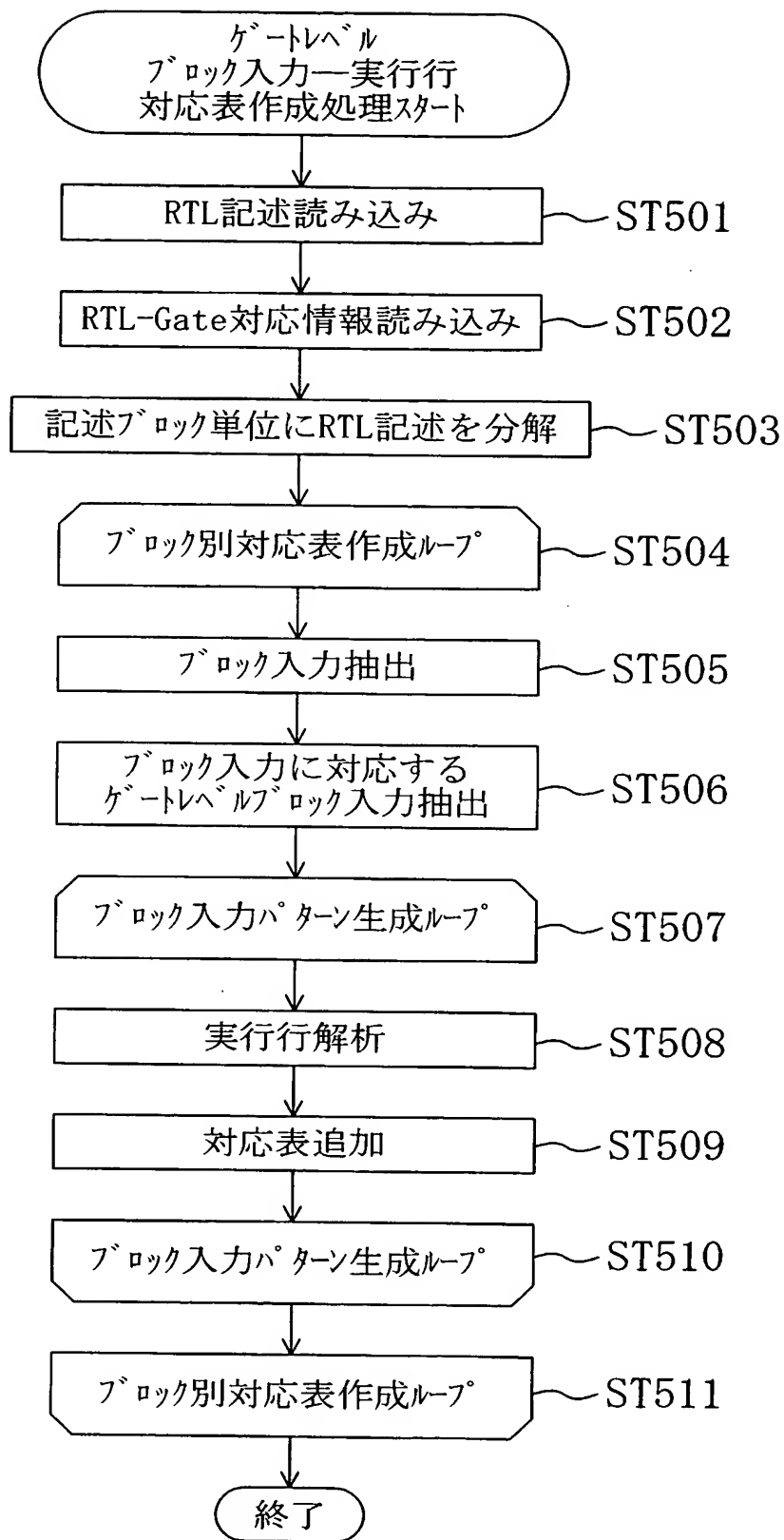
【図 14】



【図 1 5】

| RTL | Gate |
|-----|------|
| a | W1 |
| b | W2 |
| c | W4 |
| out | W7 |

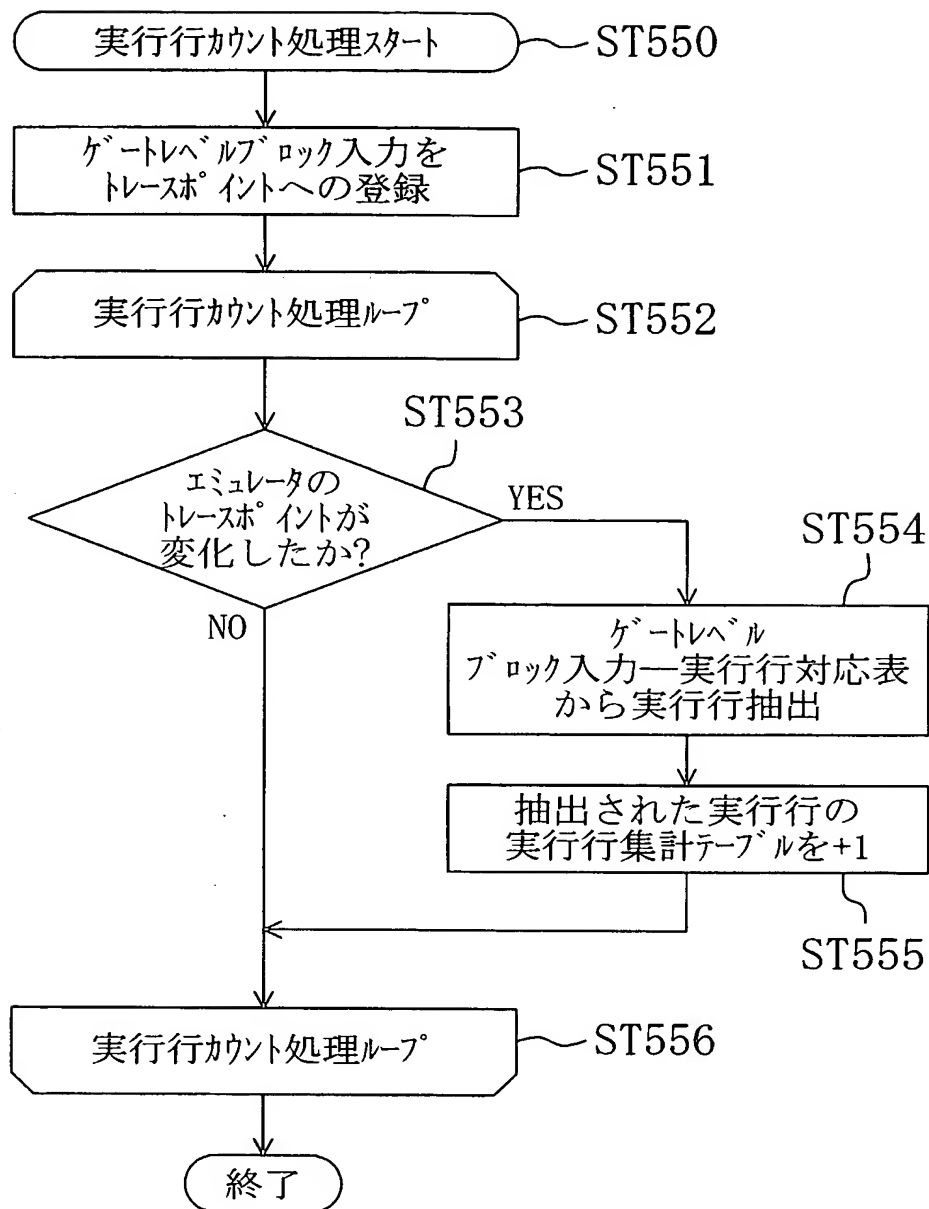
【図 16】



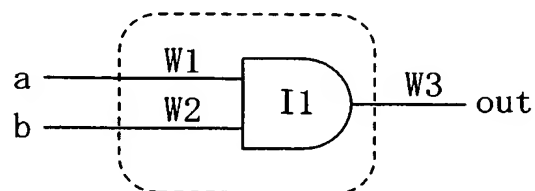
【図 17】

| 100-110行の記述ブロック | | | |
|-----------------|----|----|---------------|
| W1 | W2 | W4 | 実行行 |
| 0 | 0 | 0 | 100, 101, 102 |
| 0 | 0 | 1 | 100, 101, 103 |
| 0 | 1 | 0 | 100, 101, 104 |
| 0 | 1 | 1 | 100, 101, 105 |
| 1 | 0 | 0 | 100, 101, 106 |
| 1 | 0 | 1 | 100, 101, 107 |
| 1 | 1 | 0 | 100, 101, 108 |
| 1 | 1 | 1 | 100, 101, 109 |
| 113-115行の記述ブロック | | | |
| W4 | | | 実行行 |
| 0 | | | 113, 114 |
| 1 | | | 113, 114 |

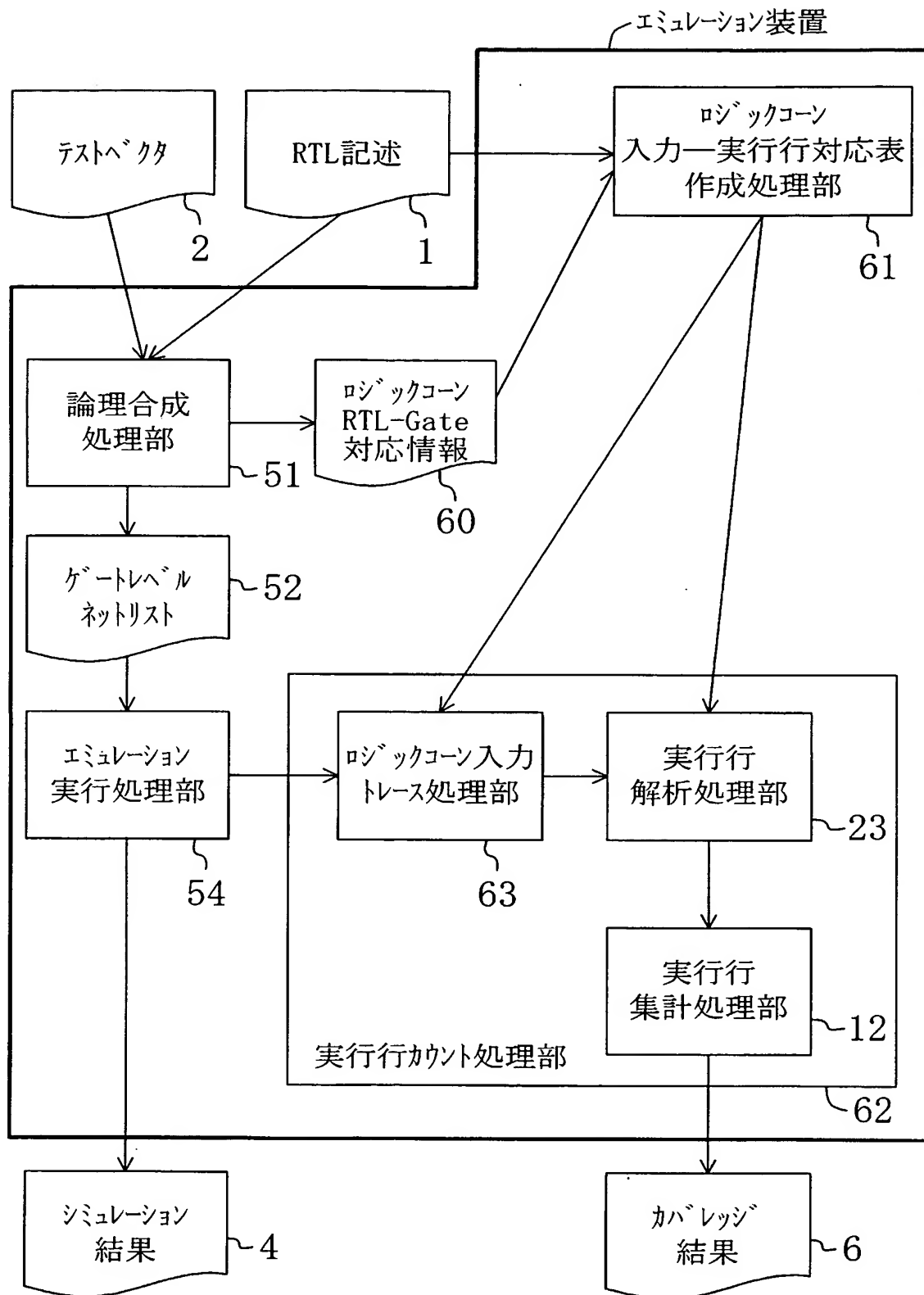
【図18】



【図19】



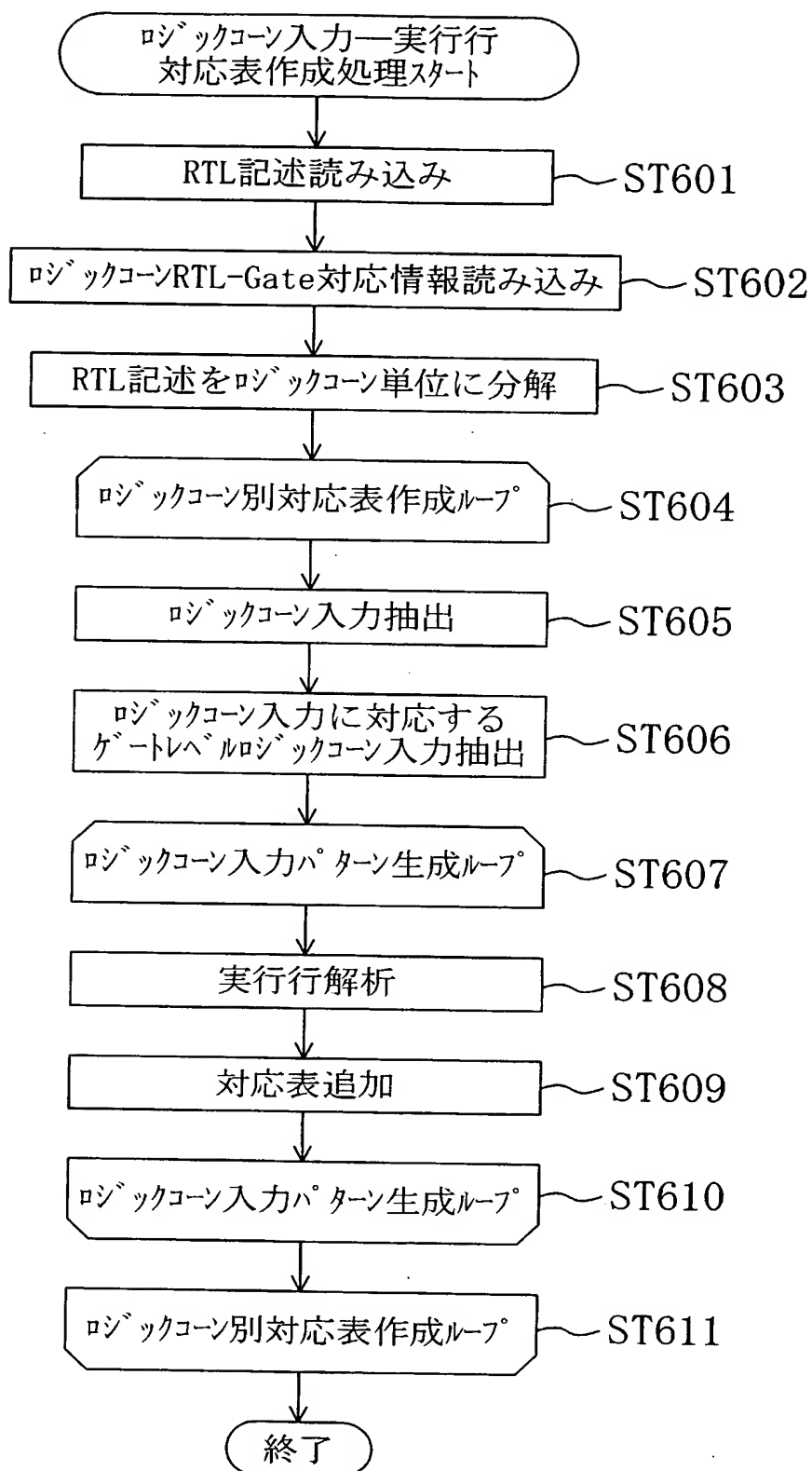
【図 20】



【図 2 1】

| RTL | Gate |
|-----|------|
| a | W1 |
| b | W2 |

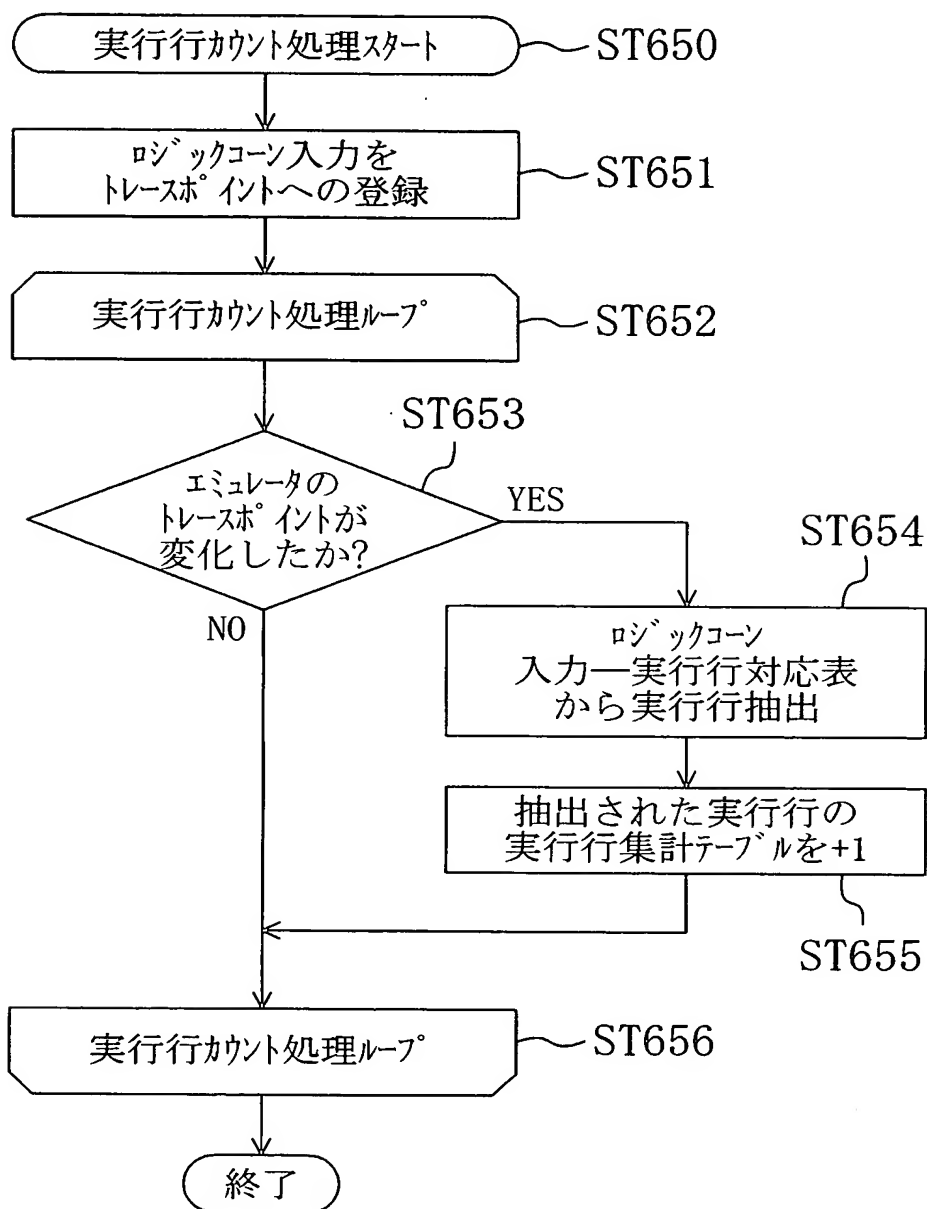
【図 22】



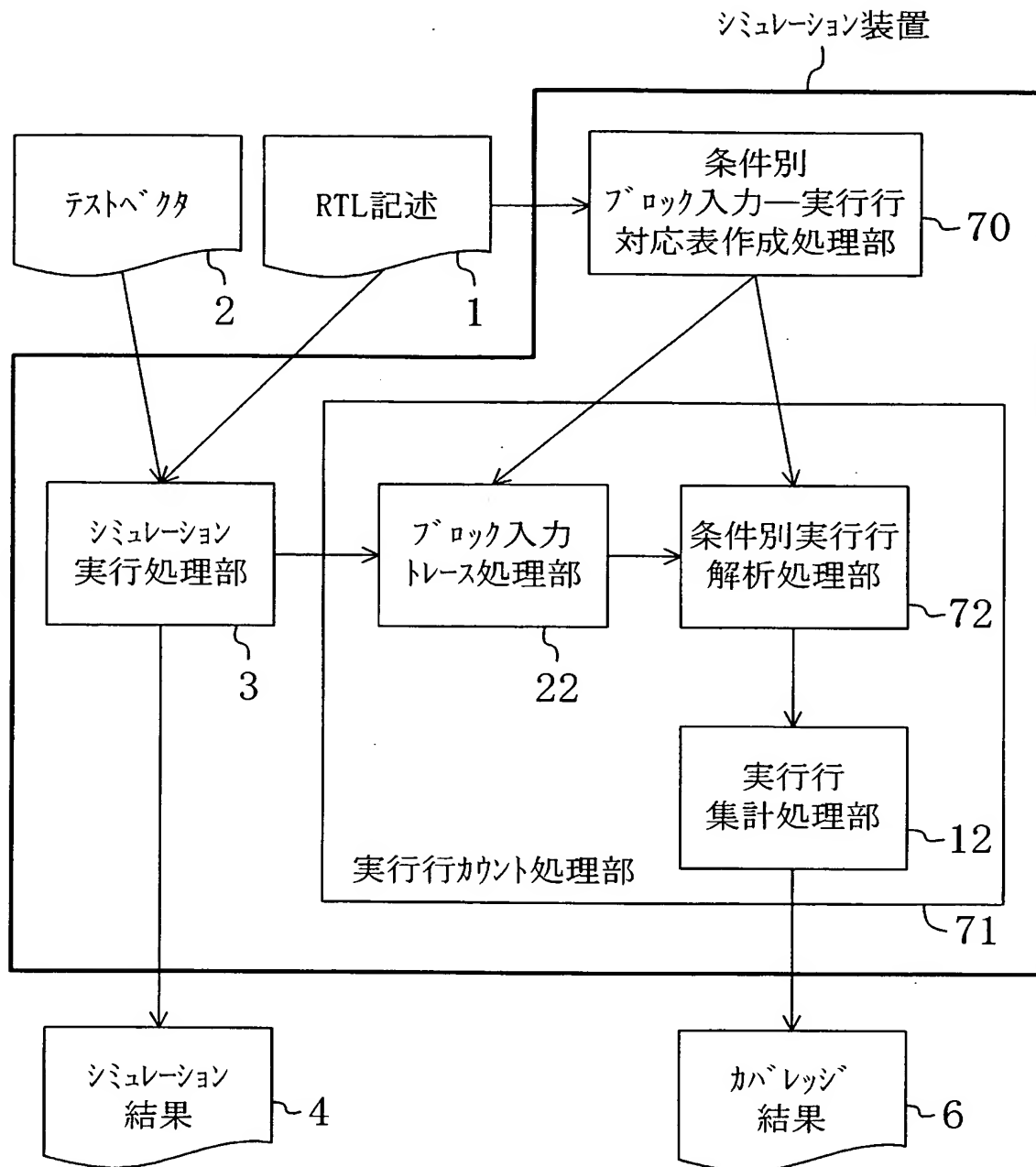
【図 23】

| W1 | W2 | 実行行 |
|----|----|-------------------------|
| 0 | 0 | 100, 101, 103, 113, 114 |
| 0 | 1 | 100, 101, 104, 113, 114 |
| 1 | 0 | 100, 101, 106, 113, 114 |
| 1 | 1 | 100, 101, 108, 113, 114 |

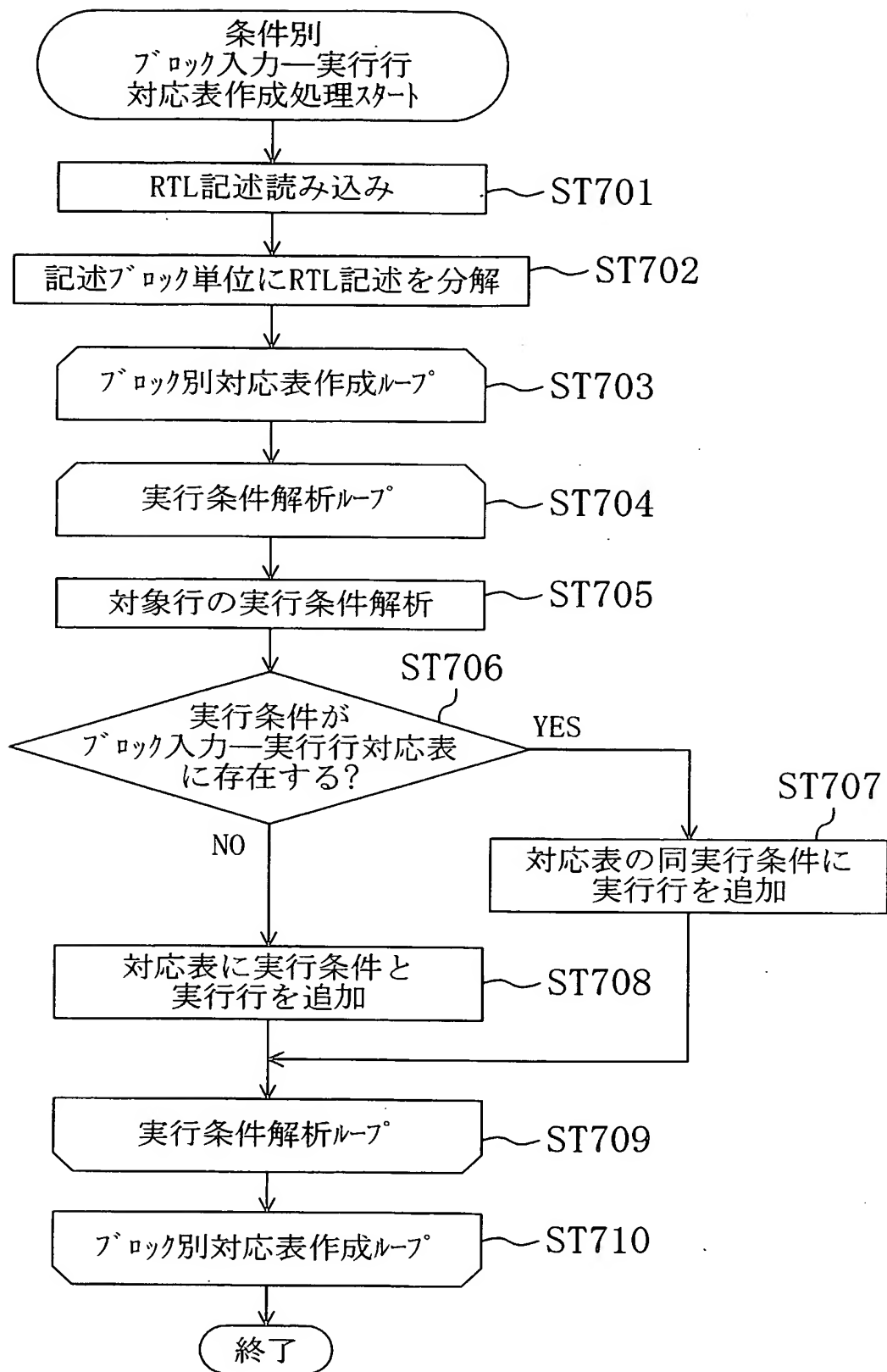
【図 24】



【図 25】



【図 26】



【図 2 7】

1
}

```

300:    always @ (data or c) begin
301:        if(data==3) out=1;
302:        else if(data==2 && c==1) out=1;
303:        else out=0;
304:    end
305:
306:    always @ (data) begin
307:        c=~data[1];
308:    end

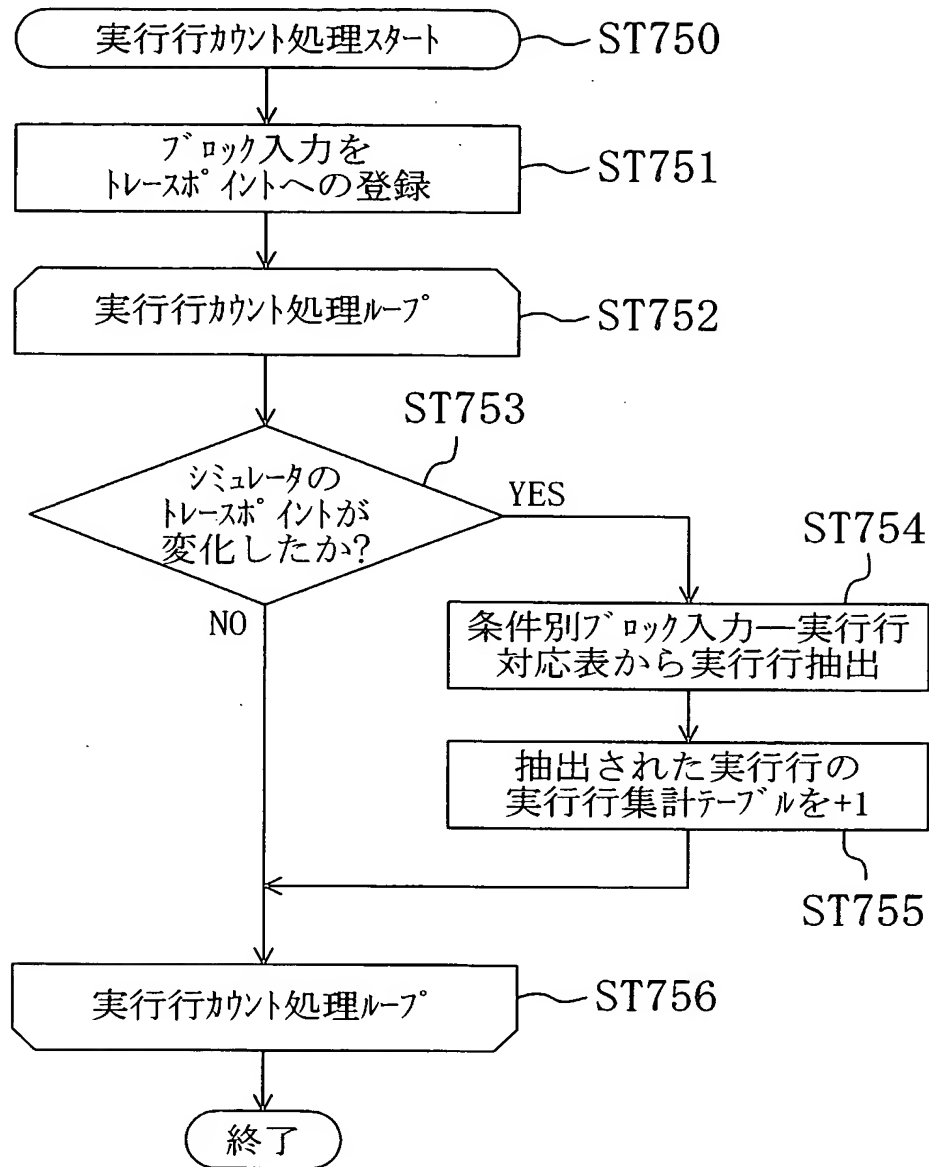
```

【図 2 8】

| 300-304行の記述ブロック | |
|----------------------------|----------|
| 入力条件 | 実行行 |
| data=3 | 300, 301 |
| !(data=3)&(data=2)&(c=1) | 300, 302 |
| !(data=3)&!(data=2)&!(c=1) | 300, 303 |

| 306-308行の記述ブロック | |
|-----------------|----------|
| 入力条件 | 実行行 |
| ALL | 306, 307 |

【図 29】



【図 30】

2

```
400:    initial begin
401:        #0      data=0;
402:        #100    data=2;
403:    end
```

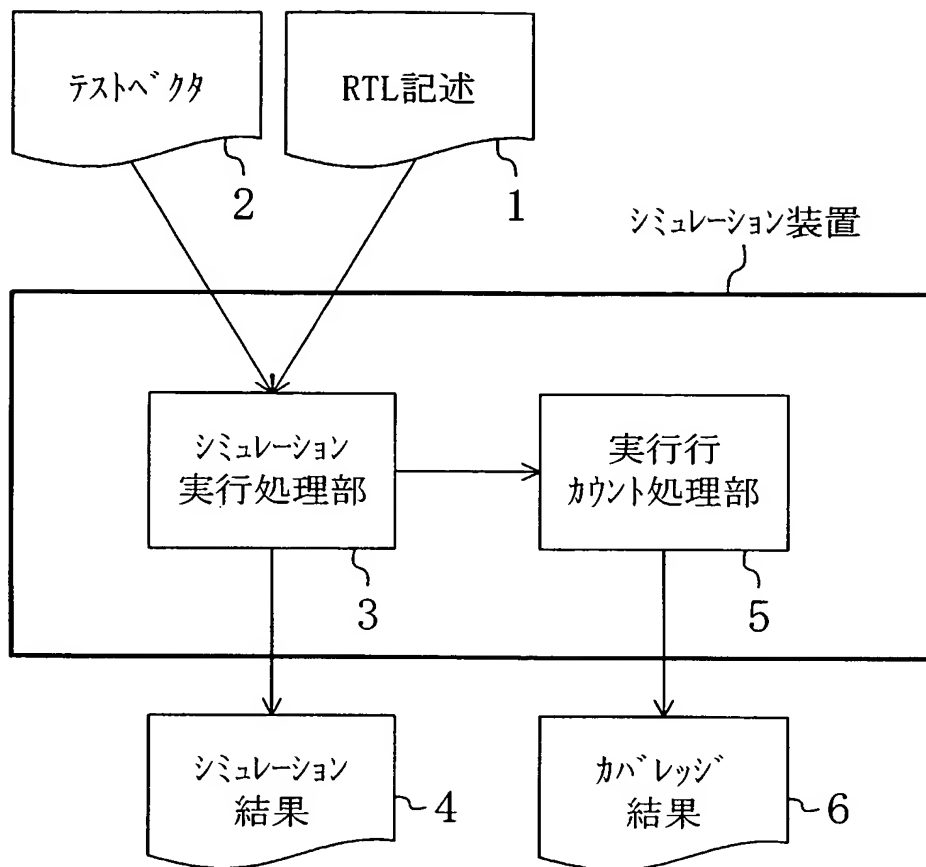
【図 31】

実行回数

6

```
4   300:    always @ (data or c) begin
      301:        if(data==3) out=1;
1   302:        else if(data==2 && c==1) out=1;
3   303:        else out=0;
      304:    end
      305:
2   306:    always @ (data) begin
2   307:        c=~data[1];
      308:    end
```

【図 3 2】



【図 3 3】

実行回数

```
4 100:    always @ (a or b or c) begin
4 101:        case({a, b, c})
      102:            3' b000:out=0;
1 103:            3' b001:out=0;
      104:            3' b010:out=0;
      105:            3' b011:out=0;
1 106:            3' b100:out=0;
1 107:            3' b101:out=1;
      108:            3' b110:out=1;
      109:            3' b111:out=1;
      110:        endcase
      111:    end
      112:
2 113:    always @ (a) begin
2 114:        c=~a;
      115:    end
```

【書類名】 要約書

【要約】

【課題】 ハードウェア記述言語検証時のソースコードカバレッジ測定において、シミュレーション実行処理が逐次処理である為、機能的に実行されないはずの行が瞬間的に実行され、不当に網羅性の高いカバレッジ結果が出力されるという問題があった。

【解決手段】 ハードウェア記述の記述ブロック単位に実行履歴を記録し、同一時刻に同じ記述ブロックが実行された場合、同一時刻の前の実行履歴を消去する工程を有し、カバレッジ測定時の誤カウントを防止する。

【選択図】 図1

特願 2 0 0 2 - 3 5 5 8 1 9

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 5 8 2 1]

1. 変更年月日

1 9 9 0 年 8 月 2 8 日

[変更理由]

新規登録

住 所

大阪府門真市大字門真 1 0 0 6 番地

氏 名

松下電器産業株式会社